Markt&Technik

Paul Lukowicz
Olaf Pfeiffer

AMIGA DATENSTRUKTUREN-LEXIKON

Das Nachschlagewerk für alle Programmierer

Auflistung aller System-Datenstrukturen: Offsets, C-Format, Modula-2-Format. Wichtige Tips auch für Basic- und Assembler-Programmierer.

Amiga Datenstrukturen-Lexikon

Amiga Datenstrukturen-Lexikon

Paul Lukowicz
Olaf Pfeiffer

Das Nachschlagewerk für alle Programmierer

Auflistung aller System-Datenstrukturen: Offsets, C-Format, Modula-Format. Wichtige Tips auch für Basic- und Assembler-Programmierer.

Markt&Technik Verlag AG

CIP-Titelaufnahme der Deutschen Bibliothek

Lukowicz, Paul:
Amiga-Datenstrukturen-Lexikon : das Nachschlagewerk
für alle Programmierer ; Auflistung aller System-Datenstrukturen:
Offsets, C-Format, Modula-Format ; wichtige Tips auch für Basic- und
Assembler-Programmierer / Paul Lukowicz ; Olaf Pfeiffer. –
Haar bei München : Markt-u.-Technik-Verl., 1990
ISBN 3-89090-250-2
NE: Pfeiffer, Olaf

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

93 92 91 90

ISBN 3-89090-250-2

Einbandgestaltung: Grafikdesign Heinz Rauner
Dieses Produkt wurde mit Desktop-Publishing-Programmen erstellt
und auf der Linotronic 300 belichtet.
Druck: Pustet, Regensburg
Printed in Germany

Inhaltsverzeichnis

VORWORT

Mit diesem Buch halten Sie kein »Lesebuch« im eigentlichen Sinn in den Händen. Vielmehr möchten wir es als ein Nachschlagewerk verstanden wissen, das immer in greifbarer Nähe zum Amiga stehen muß. Wer viel auf dem Amiga programmiert und dabei seine Möglichkeiten einigermaßen effizient ausnutzen will, kommt um eine häufige Benutzung der Systemdatenstrukturen nicht herum.

Bisher galt es hier Berge von Büchern zu wälzen, um im Endeffekt dann doch auf altbewährte Hackermethoden zurückzugreifen. Insbesondere dabei unterzog das sensitive Multitasking-System des Amiga die drei Reset-Tasten einem Dauerhärtetest! Da der Zuliefermarkt für Reset-Tasten schon seit langem überlastet ist, schrieben wir in weiser Vorraussicht dieses Buch.

Uns schwebte von Anfang an ein Werk vor, das geballte und umfassende Informationen zur Anwendung der Systemdatenstrukturen in allen gängigen Programmiersprachen (»C«, Modula-2, Assembler und Basic) enthält. Natürlich gehört dazu mehr als eine reine Auflistung der Datenstrukturen und ihrer Komponenten. Denn die wahre Funktion und Bedeutung einer Datenstruktur wird meist nur anhand ihrer Stellung im Gesamtsystem verständlich. Darüber hinaus existieren noch eine Fülle von Konstanten, Flaggen und Befehlen, die man beim Umgang mit Datenstrukturen kennen muß. Unser Buch läßt hier kaum Fragen offen!

Das größte Problem bei der Beschreibung der einzelnen Datenstrukturen bestand darin, Nützliches von Überflüssigem zu trennen. Wir hoffen nun, daß die unserer eigenen Programmiererfahrung entspringende Auswahl auch Sie überzeugt und den Guru von Ihnen fernhält.

Abschließend danken wir dem »ungenannten Dritten«, der bei der Erstellung dieses Buches »ernsthaft« mitgemischt hat!

Paul Lukowicz und Olaf Pfeiffer, im Januar 1990

1 EINLEITUNG

Voraussetzung für das Einsetzen des »Stoffes« auf den folgenden Seiten ist, daß Sie ein Amiga-Programmierer sind, der nicht nur an der »Programmier-Oberfläche« kratzt und sich mit dem Durchschnittlichen zufrieden gibt. Gehören Sie also zu denjenigen, die bestrebt sind, ihren Amiga auszureizen, dann dürften Ihnen ja auch die verwendeten drei großen »A« (Anglo-Amerikanischen Amigazismen) keine Probleme bereiten. Je weiter man in die Tiefen des Amiga vordringt, desto schwieriger wird es, diese zu umgehen, da sämtliche Definitionen und Kürzel (wie sollte es anders sein?) aus englischen Begriffen abgeleitet sind. Würde man versuchen, alles zu übersetzen, so käme man sehr schnell an den Punkt, wo nur noch Übersetzungen wie »Ursprungsebenen-Insektenvernichtungsmittel« (Source Level Debugger) möglich sind. Auf solche verzichten wir daher lieber gleich und halten uns da, wo wir es für verständlicher halten, an die entsprechenden »AAA«s. Stoßen Sie dennoch auf einen Begriff, der Ihnen nichts sagt, so schlagen Sie im Glossar nach. Rein theoretisch sollten dort alle »AAA«s (abgesehen von Begriffen wie Screen und Window) kurz erläutert sein.

Bei der Fülle der Datenstrukturen des Betriebssystems ist es nicht möglich, die Funktion aller Felder einer jeden Datenstruktur durch Ausprobieren »herauszuhacken«. Daher konnten wir uns bei der Beschreibung nicht nur auf unsere Amiga-Programmiererfahrung verlassen, sondern mußten auch anderen Quellen (siehe Literaturverzeichnis) blind vertrauen. Obwohl wir vorwiegend die Original-Commodore-Reference-Manuals sowie dokumentierte Listings verwendet haben, ist nicht auszuschließen, daß wir dabei eventuell vorhandene Fehler übernommen haben. An manchen Stellen, wo weder eine Beschreibung noch ein Kommentar in den Source-Codes aufzutreiben war, waren wir auf Vermutungen angewiesen, die dem Komponentennamen und/oder unserem Systemverständnis entsprangen. Dabei handelt es sich in erster Linie um »extrem« systeminterne Parameter, die für den Programmierer so gut wie bedeutungslos sind. Stoßen Sie also in diesem Buch auf Begriffe wie »vermutlich«, »könnte« oder »hat wahrscheinlich mit ... zu tun«, dann ist dies nur als ein Hinweis und nicht als Tatsache zu betrachten. Wo nicht einmal Vermutungen möglich waren, haben wir als Zeichen unserer Ratlosigkeit drei Fragezeichen gesetzt. Für lehrreiche Hin- und Beweise sind wir jederzeit dankbar.

Nun noch ein paar Worte zum Aufbau dieses Buches: Die Datenstrukturen sind nach Libraries und Devices in Kapitel zusammengefaßt und innerhalb dieser alphabetisch geordnet. Die Beschreibung einer Datenstruktur hat immer den gleichen Aufbau. Zunächst erfolgt jeweils eine komplette Auflistung der Datenstruktur. Von links nach rechts finden Sie dabei hintereinander den hexadezimalen und dezimalen Offset, das C-Format und abschließend das Modula-2-Format einer jeden Komponente der Datenstruktur. In der Kopfleiste steht zusätzlich hinter dem »C« das Include-File, in dem diese Datenstruktur definiert ist, und hinter Modula-2 der Name des Moduls, aus dem Modula-2-Programmierer diese Datenstruktur importieren können. Weicht das Assembler-Format der Datenstruktur in irgendeiner Form von der C-Version ab, so stehen diese Abweichungen direkt unter der Datenstruktur näher beschrieben. Vor der Beschreibung folgen noch die Referenzlisten. Aus der Datenstrukturenreferenz können Sie ersehen, welche Datenstrukturen diese Datenstruktur entweder selbst oder zumindest einen Adreßzeiger auf diese enthalten. Aus der Routinenreferenz können Sie erkennen, welche Systemroutinen auf diese zugreifen, wobei diese Listen natürlich keinen Anspruch auf Vollständigkeit erheben. Nach der Beschreibung der Datenstruktur als Gesamtes folgt eine nochmalige Auflistung aller Parameter im C-Format mit Angabe der Offsets und einer ausführlichen Beschreibung. Dabei versuchen wir, alle vordefinierten

Konstanten und deren Bedeutung anzugeben. Falls die Bedeutung einer Komponente von dem Wert einer anderen abhängt (z.B. bei den Device-Befehlen), haben wir grundsätzlich alle Möglichkeiten angegeben. Nehmen wir in der Beschreibung Bezug auf andere Komponenten, so geschieht dies ebenfalls im »C«-Format. Modula-2-Programmierer können »Ihre« Syntax der Komponente aus der Auflistung der Datenstruktur entnehmen. Maßeinheiten gibt es eigentlich nur zwei, nämlich Pixel (oder Punkte) für Größenbeschreibungen am Bildschirm und Bytes für Speicherbereichangaben.

Da die Datenstrukturen der Hardware-Library sich weitgehend auf Hardwareregister beziehen, haben wir auch diese ausführlich beschrieben. Somit beinhaltet dieses Buch zusätzlich eine recht genaue Hardwarebeschreibung.

2 Hinweise zur Benutzung der Datenstrukturen

Um auf sprachspezifische Probleme genügend einzugehen, folgen nun vier Abschnitte, in denen auf Besonderheiten der Sprachen C, Modula-2, Assembler und Basic gesondert hingewiesen wird. Um Ihnen einen direkten Vergleich zu ermöglichen, haben wir ein kurzes Beispielprogramm geschrieben und es an alle vier Programmiersprachen angepaßt. Das Programm »HalbHellDemo« soll folgendes tun:

1. Öffnen eines Bildschirms im Extra-Halfbright-Modus (64 Farben).
2. Öffnen eines Fensters auf diesem Bildschirm. Dieses Fenster soll die untere Bildschirmhälfte einnehmen und nur über ein Close-Gadget verfügen (es kann also nicht verkleinert, vergrößert oder verschoben werden).
3. Zeichnen von 62 Rechtecken, wobei 31 Rechtecke im Fenster mit den Farben 1 bis 31 und 31 Rechtecke auf dem Bildschirm mit den Farben 33 bis 63 gezeichnet werden sollen (die Farbregister werden dabei nicht mit Werten belegt, es werden die Default-Farben verwendet).
4. Warten auf das Anklicken des Close-Gadgets.
5. Schließen des Fensters und des Bildschirms.

Wie Sie wissen, wird zum Zeichnen die *RastPort*-Datenstruktur des Displayelementes benötigt, in dem gezeichnet werden soll. Das Programm »HalbHellDemo« ist insofern interessant, als der zum Bildschirm gehörige Rastport in der *Screen*-Datenstruktur eingebunden, also selbst enthalten ist. Die zum Fenster gehörende *Rastport*-Datenstruktur ist dagegen nicht in der *Window*-Datenstruktur eingebunden. In ihr steht nur ein Adreßzeiger auf die beim Zeichnen verwendete RastPort-Datenstruktur. Die verschiedenen Versionen von »HalbHellDemo« finden Sie in den folgenden vier Abschnitten.

2.1 Hinweise für C-Programmierer

Da fast das gesamte Amiga-Betriebssystem in C geschrieben ist, bereitet die Anwendung der Systemdatenstrukturen in dieser Sprache keine Schwierigkeiten. Die hier aufgelisteten Datenstrukturen entstammen den Include-Files des Aztec-C-Compilers, sie unterscheiden sich aber nicht von den Original-Include-Files von Commodore. Falls Sie einen anderen Compiler benutzen (z.B. Lattice-C oder einen der PD-Compiler), kann es also höchstens vereinzelt Probleme mit dem Aufbau der Include-Directory geben. Sie müssen sich dann der Anleitung Ihrers Compilers bedienen, um die gesuchte Datenstruktur zu finden.

Bei den meisten Flags können neben der von uns aufgelisteten Wertkonstanten auch noch die Bitnummerkonstanten verwendet werden. Der Name dieser Bitnummerkonstanten folgt aus dem Namen der entsprechenden Wertkonstanten durch Einsetzen eines »B« am Ende des Präfix (vor dem ersten Underbar »_«). Falls dort schon ein »F« (für Flag) steht, muß dieses gelöscht werden.

```
/* Programm HalbHellDemo  */

#include"intuition/intuition.h"          /* Benötigte Include-Files */
#include"graphics/gfxbase.h"
#include"graphics/view.h"

struct IntuitionBase *IntuitionBase;     /* Verwendete Datenstrukturen */
struct GfxBase *GfxBase;
struct Screen *OurScreen;
struct Window *OurWindow;
struct RastPort *ScrRPort, *WinRPort;

struct NewScreen OurNewScreen =          /* NewScreen-Struktur anlegen */
{                                  0,  0, /* LeftEdge, TopEdge */
                                 320,256, /* Width, Height */
                                       6, /* Depth */
                                   0,  1, /* DetailPen, BlockPen */
                        EXTRA_HALFBRITE, /* ViewModes */
                           CUSTOMSCREEN, /* ScreenFlags */
                                   NULL, /* Font  */
 "Extra-Halfbrite: Die Farben 32 - 63", /* Title */
                             NULL, NULL  /* Gadgets, CustomBitmap */
};

struct NewWindow OurNewWindow =          /* NewWindow-Struktur anlegen */
{                               0,  126, /* LeftEdge, TopEdge */
                                320,130, /* Width, Height */
                                  0,  1, /* DetailPen, BlockPen */
                            CLOSEWINDOW, /* IDCMPFlags */
            WINDOWCLOSE|SIMPLE_REFRESH, /* WindowFlags */
                            NULL, NULL, /* FirstGadget, Checkmark*/
        "Die Farben 0 - 31 im Fenster", /* Title */
                            NULL, NULL, /* Screen, BitMap */
                          0,0,320,256, /* Min-, MinHeigth, Max-, MaxHeight */
                           CUSTOMSCREEN  /* Type */
};


 main ()
 {
   int x1, x2, y1, y2, farbe;

   /* Graphics-Library öffnen */
   GfxBase = (struct GfxBase *) OpenLibrary("graphics.library",0);
   if (GfxBase == NULL)
    exit(FALSE);

   /* Intuition-Library öffnen */
   IntuitionBase = (struct IntuitionBase *) OpenLibrary("intuition.library",0);
   if (GfxBase == NULL)
    exit(FALSE);
```

```
  /* Screen Öffnen */
  OurScreen = (struct Screen *) OpenScreen(&OurNewScreen);
  if (OurScreen == NULL)
   exit(FALSE);

  OurNewWindow.Screen = OurScreen;     /* Adreßzeiger auf Screen holen */

  /* Window Öffnen */
  OurWindow = (struct Window *) OpenWindow(&OurNewWindow);
  if (OurWindow == NULL)
   exit(FALSE);

  ScrRPort = &(OurScreen->RastPort);   /* RastPort ist in Screen-Struktur ein-
                                          gebunden, also Adresse berechnen. */
  WinRPort = OurWindow->RPort;         /* Window-Struktur enthält Zeiger auf
                                          RastPort, also Adresse kopieren. */

  farbe = 0;
  x1 =  5; y1 = 15;                    /* Startwerte für Rechtecke festlegen */
  x2 = 55; y2 = 55;
  do                                   /* Zeichenschleife (alle Farben) */
  {
    x1+=7; y1++; x2+=8; y2+=2;
    SetAPen(WinRPort,++farbe);         /* Zeichenfarbe für Fenster setzen */
    RectFill(WinRPort,x1,y1,x2,y2);    /* Rechteck im Fenster zeichnen */
    SetAPen(ScrRPort,farbe+32);        /* Zeichenfarbe für Screen setzen */
    RectFill(ScrRPort,x1,y1,x2,y2);    /* Rechteck im Screen zeichnen */
  } while (farbe < 31);                /* alle Farben durch ? */

  WaitPort(OurWindow->UserPort);       /* Auf Close-Gadget warten */
  CloseWindow(OurWindow);              /* Fenster schließen */
  CloseScreen(OurScreen);              /* Screen schließen */
  CloseLibrary(GfxBase);               /* Libraries schließen */
  CloseLibrary(IntuitionBase);
}
```

2.2 Hinweise für Modula-2-Programmierer

Die in diesem Buch vorhandenen Auflistungen der Datenstrukturen im Modula-2-Format entsprechen dem, das der M2Amiga-Compiler Version 3.3d der A+L AG verwendet. Eine Besonderheit dieses Compilers besteht in der Handhabung der Libraries. Normalerweise ist ein Zugriff auf eine Library, abgesehen von der Exec-Library, nur möglich, wenn diese zuvor durch die Routine *OpenLibrary* der Exec-Library geöffnet wurde. Nach Beendigung aller Zugriffe erfolgt dann eine Schließung der Library durch die *CloseLibrary*-Routine. Der M2Amiga-Compiler nimmt uns diese Arbeit ab, da er durch die IMPORT-Listen nicht nur erkennt, welche Libraries geöffnet werden müssen, sondern das Öffnen und Schließen auch selbst übernimmt. An dieser Stelle sei auch gesagt, daß wir bei dem Adreßzeiger NULL nicht jedesmal das Modula-2-Format (NIL) dazuschreiben. Falls Sie es also noch nicht wissen sollten, merken Sie sich bitte: »C«-NULL ≅ »Modula-2«-NIL.

Betreffend der Datenstrukturen gilt grundsätzlich, daß neben dem eigentlichen Datenverbund auch ein Adreßzeiger auf ihn definiert ist. Der Name des Adreßzeigers ist fast der gleiche wie der der zugehörigen Datenstruktur, es wird lediglich die Endung *Ptr* angehängt (z.B.: *Window*, *WindowPtr*).

Die Definition der Flags ist ebenfalls immer einheitlich: In einer Datenstruktur finden Sie den *BITSET*-Typ, den Sie daran erkennen, daß er die Endung »*Set*« hat. Um einzelne Flags setzen zu können, muß der zugehörige Aufzählungstyp ebenfalls importiert werden. Hat eine bestimmte Kombination von mehreren Bits eine gesonderte Bedeutung, so sind diese als Konstanten definiert. Beachten Sie also bitte besonders beim Setzen von Flags, ob es sich um einzelne Bits (Importieren des zugehörigen Aufzählungstyps genügt) oder Kombinationen von mehreren handelt (entsprechende Konstante muß extra importiert werden). Betrachten Sie als Beispiel dazu die *NewWindow*-Datenstruktur, in der der Datentyp *WindowFlagSet* verwendet wird. Wollen Sie einzelne Bits setzen, so reicht es, neben diesem Datentyp den zugehörigen Aufzählungstyp *WindowFlags* zu importieren. Nun hat aber darüber hinaus das gleichzeitige Setzen der beiden Flags *simpleRefresh* und *superBitMap* eine eigene Bedeutung. Die Kombination dieser beiden Flags wird daher getrennt als die Konstante *otherRefresh* definiert. Wollen Sie diese verwenden, so muß sie ebenfalls importiert werden.

Nun zu der Modula-2-Version des bereits in der Einleitung erwähnten Beispielprogrammes:

```
MODULE HalbHellDemo;


FROM SYSTEM     IMPORT ADR;           (* Funktion: Berechnet einen Adreßzeiger auf
                                         ein Datenelement *)

FROM Exec       IMPORT WaitPort;      (* Routine: Warten auf Ereignis *)
FROM Intuition  IMPORT OpenScreen,    (* Funktion: Öffnet einen Bildschirm *)
                       CloseScreen,   (* Routine: Schließt einen Bildschirm *)
                       NewScreen,     (* NewScreen-Struktur *)
                       ScreenPtr,     (* Zeiger auf Screen-Struktur *)
                       OpenWindow,    (* Funktion: Öffnet ein Fenster *)
                       CloseWindow,   (* Routine: Schließt ein Fenster *)
                       NewWindow,     (* NewWindow-Struktur *)
                       WindowPtr,     (* Zeiger auf Window-Struktur *)
                       IDCMPFlags,    (* Aufzählungstyp der IDCMP-Flags *)
                       IDCMPFlagSet,  (* BITSET-Typ der IDCMP-Flags *)
                       WindowFlags,   (* Aufzählungstyp der Window-Flags *)
                       WindowFlagSet, (* BITSET-Typ der Window-Flags *)
                       ScreenFlagSet, (* BITSET-Typ der Screen-Flags *)
                       customScreen;  (* Kombination der ersten vier Bits aus
                                         ScreenFlagSet, wird anstelle des Auf-
                                         zählungstypen ScreenFlags verwendet.*)

FROM Graphics   IMPORT ViewModes,     (* Aufzählungstyp der View-Modi *)
                       ViewModeSet,   (* BITSET-Typ der View-Modi *)
                       RastPortPtr,   (* Zeiger auf RastPort-Struktur *)
                       SetAPen,       (* Routine: Zeichenfarbe setzen *)
                       RectFill;      (* Routine: Rechteck zeichnen *)

CONST ScrTitle = "Extra-Halfbrite: Die Farben 32 - 63";
      WinTitle = "Die Farben 0 - 31 im Fenster";
```

```
VAR OurNewScreen : NewScreen;          (* NewScreen-Struktur *)
    OurScreen    : ScreenPtr;          (* Zeiger auf Screen-Struktur *)
    OurNewWindow : NewWindow;          (* NewWindow-Struktur *)
    OurWindow    : WindowPtr;          (* Zeiger auf Window-Struktur *)
    ScrRPort     : RastPortPtr;        (* Zeiger auf RastPort des Screens *)
    WinRPort     : RastPortPtr;        (* Zeiger auf RastPort des Windows *)
    x1,x2        : [0..320];           (* Die obere linke und untere Rechte *)
    y1,y2        : [0..256];           (* Ecke der zu zeichnenden Rechtecke. *)
    farbe        : [0..63];            (* Nummer eines Farbregisters *)


BEGIN

  WITH OurNewScreen DO                 (* NewScreen-Struktur anlegen *)
    leftEdge     := 0;
    topEdge      := 0;
    width        := 320;
    height       := 256;
    depth        := 6;
    detailPen    := 0;
    blockPen     := 1;
    viewModes    := ViewModeSet{extraHalfbrite};
    type         := customScreen;
    font         := NIL;
    defaultTitle := ADR(ScrTitle);
    gadgets      := NIL;
    customBitMap := NIL;  END; (* WITH OurNewScreen DO *)

  OurScreen := OpenScreen(OurNewScreen);   (* Screen öffnen *)

  IF OurScreen <> NIL THEN                 (* War Öffnen erfolgreich? *)

(****  Da die RastPort-Struktur in der Screen-Struktur eingebunden ist,    ****)
(****  muß seine Adresse durch ADR() berechnet werden.                     ****)
    ScrRPort := ADR(OurScreen^.rastPort);
(****                                                                       ****)

    WITH OurNewWindow DO                   (* NewWindow-Struktur anlegen *)
      leftEdge     := 0;
      topEdge      := 126;
      width        := 320;
      height       := 130;
      detailPen    := 0;
      blockPen     := 1;
      idcmpFlags   := IDCMPFlagSet{closeWindow};
      flags        := WindowFlagSet{windowClose,simpleRefresh};
      firstGadget  := NIL;
      checkMark    := NIL;
      title        := ADR(WinTitle);
      screen       := OurScreen;
```

2

```
        bitMap       := NIL;
        minWidth     := 0;
        minHeight    := 0;
        maxWidth     := 320;
        maxHeight    := 256;
        type         := customScreen;
      END; (* WITH OurNewWindow DO *)

      OurWindow := OpenWindow(OurNewWindow); (* Fenster öffnen *)

      IF OurWindow <> NIL THEN               (* War das Öffnen erfolgreich? *)

(****  Da die Window-Struktur nur einen Zeiger auf die RastPort-Struktur   ****)
(****  enthält, muß dieser nur kopiert werden.                              ****)
        WinRPort := OurWindow^.rPort;
(****                                                                       ****)

        farbe := 0;                          (* Initialisieren der Variablen *)
        x1 :=  5; y1 := 15;
        x2 := 55; y2 := 55;

        REPEAT                               (* Schleife für Rechtecke *)
          INC(farbe);                        (* Farbregisternummer erhöhen *)
          INC(x1,7);                         (* Neue obere linke Ecke und *)
          INC(y1);                           (* neue untere rechte Ecke des *)
          INC(x2,8);                         (* aktuellen Rechtecks berechnen. *)
          INC(y2,2);
          SetAPen(WinRPort,farbe);           (* Window-Zeichenfarbe setzen *)
          RectFill(WinRPort,x1,y1,x2,y2);    (* Rechteck im Window zeichnen *)
          SetAPen(ScrRPort,farbe+32);        (* Screen-Zeichenfarbe setzen *)
          RectFill(ScrRPort,x1,y1,x2,y2);    (* Rechteck im Screen zeichnen *)
        UNTIL farbe = 31;                    (* Alle Farben von 1 - 31 durch? *)
        WaitPort(OurWindow^.userPort);       (* Warten auf Systemnachricht des
                                                Fensters *)
        CloseWindow(OurWindow);              (* Fenster schließen *)
      END; (* IF OurWindow <> NIL *)

      CloseScreen(OurScreen);                (* Bildschirm schließen *)
    END; (* IF OurScreen <> NIL *)

END HalbHellDemo.
```

Den Modula-2-Programmierern empfehlen wir an dieser Stelle dringend, das mächtige Modul-Konzept von Modula-2 auch auszunutzen und nicht immer wieder erneut Räder zu erfinden. So gibt es bereits einige Amiga-spezifische Modul-Bibliotheken wie die Amiga-Treasures oder File-Treasures, mit denen das aufwendige Hantieren mit den Datenstrukturen (fast) entfällt. Das Erstellen solcher Implementations-Module sollte das Ziel eines jeden Modula-2-Programmierers sein, um später bei nochmaligem Gebrauch auf bereits Erarbeitetes sofort zurückgreifen zu können.

2.3 Hinweise für Assembler-Programmierer

In diesem Abschnitt wollen wir nur auf einige typische Beispiele (sowie Tücken und Gefahren) aus dem Leben eines Assembler-Programmierers hinweisen. Er soll einen allgemeinen Überblick über Besonderheiten liefern, die es im Zusammenhang mit der Assembler-Programmierung auf dem Amiga zu beachten gilt. Wir beziehen uns bei den folgenden Angaben immer auf den Devpac-Assembler von HiSoft.

Zunächst einmal zu so allgemeinen Dingen wie dem Öffnen und Schließen der Libraries und Devices. In Assembler sieht die Vorgehensweise wie folgt aus:

```
OpenLib  lea      LibName,a1         ; Adreßzeiger auf den Namen der Library.
         move.l   #0,d0              ; Versionsnummer der Library (0: No. egal).
         CALLEXEC OpenLibrary        ; Öffnen der Library.
         tst.l    d0                 ; Vergleichen des Adreßzeigers mit 0.
         beq      Exit               ; d0 = 0 : Das Öffnen war nicht erfolgreich.
         move.l   d0,_IntuitionBase  ; Adreßzeiger auf die geöffnete Library.

                                     ; Hier kann das eigentliche Programm stehen.

CloseLib move.l   IntuiZg,a1         ; Adreßzeiger auf die zu schließende Library.
         CALLEXEC CloseLibrary       ; Library schließen.

Exit     rts                         ; Programm verlassen.

LibName        dc.b "intuition.library",0 ; Name der Library.
_IntuitionBase dc.l 0                     ; Adreßzeiger auf die Library.
```

Um zum eigentlichen Thema dieses Buches zu kommen, nun etwas über den Gebrauch der Datenstrukturen. In Assembler stimmt die Definition der Datenstrukturen in weiten Teilen mit dem Format der C-Definition überein. Wir haben uns daher entschlossen, das Assembler-Format nicht extra mit abzudrukken. Sie finden dafür am Ende einer jeden Datenstruktur Informationen, die Ihnen sagen, ob und, wenn ja, wie sich das Assembler-Format vom C-Format unterscheidet. Normalerweise beschränkt sich dies auf die Angabe eines Präfix. Setzen Sie dieses Präfix vor die Namen der einzelnen Komponenten im C-Format, so erhalten Sie den Namen im Assembler-Format.

Nun zu der Vorgehensweise, wie Speicherplatz für eine Datenstruktur bereits im Listing freizuhalten ist: Die Länge einer jeden Datenstruktur (in Bytes) ist jeweils zusätzlich definiert. Um anzugeben, von welcher Datenstruktur Sie die Länge haben möchten, müssen Sie das Präfix der jeweiligen Datenstruktur mit der Endung *SIZE* versehen. Um Speicherplatz für eine *BitMap*-Datenstruktur (Präfix: *bm_*) bereits im Listing zu reservieren, kann man also folgendes einsetzen:

```
OurBitMap  ds.b bm_SIZE         ; Speicherbereich der Größe einer BitMap-Daten-
                                ; struktur freihalten.
```

Abschließend das bereits in der Einleitung erwähnte Beispielprogramm, das den Umgang mit den einzelnen Komponenten der Datenstrukturen verdeutlichen soll. Das Beispiel ist so gewählt, daß einerseits auf eine in die *Screen*-Datenstruktur eingebundene *RastPort*-Datenstruktur zugegriffen wird, andererseits aber auch auf eine *RastPort*-Datenstruktur, von der in der *Window*-Datenstruktur lediglich ein Adreßzeiger steht.

```
; Programm: HalbHellDemo

        incdir"Ass:include/"                    ; Pfad zu den Include-Files.

        include exec/exec_lib.i                 ; Einbinden der benötigten
        include intuition/intuition.i           ; Include-Files.
        include intuition/intuition_lib.i
        include graphics/graphics_lib.i

OpenIntLib                                      ; Öffnen der Intuition-Library.
        clr.l     d0                            ; Versionsnummer ist egal.
        lea       IntName,a1                    ; Zeiger auf den Library-Namen.
        CALLEXEC OpenLibrary                    ; Aufruf der Systemroutine.
        tst.l     d0                            ; Öffnen erfolgreich?
        beq       ExitFinalis                   ; Nein: Abbrechen.
        move.l    d0,_IntuitionBase             ; Ja: Zeiger merken.

OpenGrafLib                                     ; Öffnen der Graphics-Library.
        clr.l     d0                            ; Versionsnummer ist egal.
        lea       GrafName,a1                   ; Zeiger auf den Library-Namen.
        CALLEXEC OpenLibrary                    ; Aufruf der Systemroutine.
        tst.l     d0                            ; Öffnen erfolgreich?
        beq       ExitCloseInt                  ; Nein: Abbrechen.
        move.l    d0,_GfxBase                   ; Ja: Zeiger merken.

OpenScr                                         ; Öffnen eines Screens.
        move.l    OurNewScreen,a0               ; Zeiger auf NewScreen-Struktur.
        CALLINT   OpenScreen                    ; Aufruf der Systemroutine.
        tst.l     d0                            ; Öffnen erfolgreich?
        beq       ExitCloseGraf                 ; Nein: Abbrechen.
        move.l    d0,OurScreen                  ; Ja: Screen-Zeiger merken.

; ****  Adreßzeiger auf den RastPort des Screens berechnen: Die RastPort-    ****
; ****  Struktur ist in der Screen-Struktur eingebunden, daher muß zu der    ****
; ****  Adresse der Screen-Struktur der Offset sc_RastPort (#84) hinzuge-    ****
; ****  gezählt werden.                                                      ****
        addi.l    #sc_RastPort,d0
; ****                                                                       ****

        move.l    d0,ScrRPort                   ; Zeiger auf RastPort merken.

OpenWin                                         ; Öffnen eines Windows.
        move.l    OurNewWindow,a0               ; Zeiger auf NewWindow-Struktur.
        move.l    OurScreen,nw_Screen(a0)       ; Screen-Zeiger in NewWindow-
                                                ; Struktur auf eigenen Screen
        CALLINT   OpenWindow                    ; setzen.
        tst.l     d0                            ; Öffnen erfolgreich?
        beq       ExitCloseScr                  ; Nein: Abbrechen.
        move.l    d0,OurWindow                  ; Ja: Window-Zeiger merken.
        move.l    d0,a0                         ; Zeiger wird in einem Adreß-
                                                ; register benötigt.
```

```
; ****  Den Adreßzeiger auf den RastPort des Window berechnen: Die Window-   ****
; ****  Struktur enthält selbst nur einen Adreßzeiger auf die RastPort-      ****
; ****  Struktur, der "nur" kopiert werden muß. Addiert man zu der Adresse   ****
; ****  der Window-Struktur den Offset wd_RPort (#50), so erhält man die     ****
; ****  Adresse, an der der gewünschte Zeiger steht.                         ****
        move.l   wd_RPort(a0),WinRPort
; ****                                                                       ****

Rects                                             ; Rechtecke ausgeben.
        addq.b   #1,farbe                         ; Nächste Farbe auswählen.
        addi.w   #7,x1                            ; Koordinaten der linken
        addi.w   #1,y1                            ; oberen Ecke setzen.
        addi.w   #8,x2                            ; Koordinaten der rechten
        addi.w   #2,y2                            ; unteren Ecke setzen.
        move.l   WinRPort,a1                      ; Zeiger auf Window-RastPort.
        move.b   farbe,d0                         ; Die nächste Zeichenfarbe
        CALLGRAF SetAPen                          ; setzen.
        move.w   x1,d0                            ; Koordinaten der linken
        move.w   y1,d1                            ; oberen Ecke holen.
        move.w   x2,d2                            ; Koordinaten der rechten
        move.w   y2,d3                            ; unteren Ecke holen.
        CALLGRAF RectFill                         ; Rechteck auf Window ausgeben.
        move.l   ScrRPort,a1                      ; Zeiger auf Screen-RastPort.
        move.b   farbe,d0                         ; Farbwert erneut holen und
        addi.b   #$20,d0                          ; 32 addieren.
        CALLGRAF SetAPen                          ; Farbe setzen.
        move.w   x1,d0                            ; Koordinaten der linken
        move.w   y1,d1                            ; oberen Ecke holen.
        move.w   x2,d2                            ; Koordinaten der rechten
        move.w   y2,d3                            ; unteren Ecke holen.
        CALLGRAF RectFill                         ; Rechteck auf Screen ausgeben.
        cmp.b    #31,farbe                        ; Alle Farben (1-31) durch?
        bne      Rects                            ; Nein: Also nochmal.

WaitForAction                                     ; Wait-State setzen (Warten
                                                  ; mit minimaler Rechenzeit).
        move.l   OurWindow,a0                     ; Zeiger auf Window holen.
        move.l   wd_UserPort(a0),a0               ; Zeiger auf UserPort (MsgPort)
                                                  ; holen.
        CALLEXEC WaitPort                         ; Warten auf Systemnachricht.

ExitLegalis                                       ; Vor Programmbeendigung
        move.l   OurWindow,a0                     ; Fenster,
        CALLINT  CloseWindow
ExitCloseScr
        move.l   OurScreen,a0                     ; Bildschirm,
        CALLINT  CloseScreen
ExitCloseGraf
        move.l   _GfxBase,a1                      ; Graphics-Library und
        CALLEXEC CloseLibrary
ExitCloseInt
        move.l   _IntuitionBase,a1                ; Intuition-Library wieder
```

```
        CALLEXEC CloseLibrary                    ; ordnungsgemäß schließen.
ExitFinalis
        rts                                      ; Programmende.

        cnop 0,2                                 ; Nachfolgende Adresse gerade?

OurNewScreen                                     ; NewScreen-Datenstruktur:
                dc.w 0                           ; LeftEdge
                dc.w 0                           ; TopEdge
                dc.w 320                         ; Width
                dc.w 256                         ; Height
                dc.w 6                           ; Depth
                dc.b 0                           ; DetailPen
                dc.b 1                           ; BlockPen
                dc.w 128                         ; ViewModes: EXTRAHALFBRIGHT
                dc.w CUSTOMSCREEN                ; Type
                dc.l 0                           ; Font
                dc.l ScrTitle                    ; Title
                dc.l 0                           ; Gadgets
                dc.l 0                           ; CustomBitMap

OurNewWindow                                     ; NewWindow-Datenstruktur:
                dc.w 0                           ; LeftEdge
                dc.w 126                         ; TopEdge
                dc.w 320                         ; Width
                dc.w 130                         ; Height
                dc.b 0                           ; DetailPen
                dc.b 1                           ; BlockPen
                dc.l CLOSEWINDOW                 ; IDCMPFlags
                dc.l WINDOWCLOSE!SIMPLE_REFRESH  ; WindowFlags
                dc.l 0                           ; FirstGadget
                dc.l 0                           ; CheckMark
                dc.l WinTitle                    ; Title
                dc.l 0                           ; Screen
                dc.l 0                           ; BitMap
                dc.w 0                           ; MinWidth
                dc.w 0                           ; MinHeight
                dc.w 320                         ; MaxWidth
                dc.w 256                         ; MaxHeight
                dc.w CUSTOMSCREEN                ; Type

x1              dc.w 5                           ; Startwerte für die linke
y1              dc.w 15                          ; obere und die rechte untere
x2              dc.w 55                          ; Ecke der zu Zeichnenden
y2              dc.w 55                          ; Rechtecke.
farbe           dc.b 0                           ; Startwert für Farbregister.

        cnop0,2                                  ; Die nachfolgende Adresse muß
                                                 ; gerade sein.

_IntuitionBase  dc.l 0                           ; Zeiger für Intuition-Library.
_GfxBase        dc.l 0                           ; Zeiger für Graphics-Library.
```

```
OurScreen       dc.l 0                      ; Zeiger für Screen-Struktur.
OurWindow       dc.l 0                      ; Zeiger für Window-Struktur.
ScrRPort        dc.l 0                      ; Zeiger für RastPort-Struktur
                                            ; des Screens.
WinRPort        dc.l 0                      ; Zeiger für RastPort-Struktur
                                            ; des Windows.

ScrTitle        dc.b "Extra-Halfbrite: Die Farben 32 - 63",0
        cnop 0,2

WinTitle        dc.b "Die Farben 0 - 31 im Fenster",0
        cnop 0,2

IntName         dc.b "intuition.library",0
        cnop 0,2

GrafName        dc.b "graphics.library",0
```

Abschließend noch ein paar warnende Worte zur Benutzung der Include-Files. In Assembler bestehen diese fast ausschließlich aus Offset-Tabellen, bei denen einer bestimmten Buchstabenfolge eine Adreßdistanz zugeordnet wird. Leider ist es beim Devpac-Assembler möglich, Systemroutinen aufzurufen, die es gar nicht gibt. Syntaktisch wird beispielsweise der Aufruf »CALLINT WritePixel« ohne Probleme (also ohne Fehlermeldung) geschluckt. Da es sich aber bei »WritePixel« nicht um eine Routine der Intuition-Library (CALLINT), sondern um eine der Graphics-Libraries (CALLGRAF) handelt, hat so ein Aufruf mehr oder weniger schwerwiegende Folgen. Je mehr Libraries geöffnet sind, desto »gefährlicher« wird die Programmierung.

2.4 Hinweise für Amiga-Basic-Programmierer

Da in Amiga-Basic heterogene Datenstrukturen (RECORD-Verbunde in Modula-2, struct-Strukturen in C) grundsätzlich unbekannt sind, erweist sich der Datenstrukturzugriff hier als äußerst umständlich: Ausgehend von der Anfangs- bzw. Basisadresse eines Datensatzes, kann auf die einzelnen Komponenten über deren Offset mittels entsprechender PEEKx- und POKEx-Anweisungen zugegriffen werden. In dieser Hinsicht ähnelt die Vorgehensweise der Assemblerprogrammierung. Allerdings gibt es für Amiga-Basic keine Include-Dateien, in denen die Offsets und Flagswerte symbolisch aufbereitet sind. Dem Amiga-Basic-Programmierer bleibt deshalb nichts anderes übrig, als diese jedesmal in numerischer Form anzugeben. Dabei wird das vorliegende Buch sicher schnell zum unentbehrlichen Hilfsmittel werden!

Zur Bereitstellung des Speicherplatzes für die von einem Amiga-Basic-Programm selbst benötigten Systemdatenstrukturen bieten sich zwei unterschiedliche Verfahren an:

1. Statische Deklaration eines entsprechend großen Zahlenfelds mittels der DIM-Anweisung.
2. Dynamische Speicherbeschaffung durch Aufruf einer Systemroutine wie *AllocMem* aus »Exec« oder *AllocRemember* aus »Intuition«.

Im weiter unten abgedruckten Amiga-Basic-Programm »HalbHellDemo« findet Methode 1 bei der Benutzung von *NewScreen*- und *NewWindow*-Datenstrukturen Verwendung. Natürlich ist in beiden Fällen umständliches POKEn zur Initialisierung der Datenstrukturen nötig. Zu beachten ist darüber hinaus die Tatsache, daß die in Systemdatenstrukturen verwendeten Zeichenketten durchweg mit einem Nullzeichen = CHR$(0) abgeschlossen werden müssen.

Wie man die Amiga-Libraries in Amiga-Basic mittels der LIBRARY-Anweisung öffnet und die einzelnen Systemroutinen danach durch CALL-Aufrufe anspringt, sollte bekannt sein. Weitere Tips und Hinweise zur Systemprogrammierung entnehmen Sie am besten Kapitel 6 des Amiga-Basic-Handbuchs von Commodore. Viel Spaß also mit dem nachfolgenden Beispielprogramm und dem gesamten Buch!

```
'---------------------------------------------------------------------------'
'
'  Benoetigte Systembibliotheken oeffnen und alle daraus verwendeten
'  Funktionen mit Ergebnistyp und Parametern deklarieren.
'
'---------------------------------------------------------------------------'


LIBRARY  "exec.library"

 DECLARE FUNCTION  WaitPort& (Port&)  LIBRARY


LIBRARY  "graphics.library"


LIBRARY  "intuition.library"

 DECLARE FUNCTION  OpenScreen& (NewScreen&)  LIBRARY
 DECLARE FUNCTION  OpenWindow& (NewWindow&)  LIBRARY


'---------------------------------------------------------------------------'
'
'  "HalbHellDemo" :  Amiga-BASIC-Version
'
'---------------------------------------------------------------------------'


ScrTitle$ = "Extra-Halfbrite: Die Farben 32 - 63" + CHR$(0)

EXTRAHALFBRITE% = 128   ' Flagswert fuer Extra-Halfbrite Modus
CUSTOMSCREEN%   =  15   ' Typwert fuer Benutzer-definierten Bildschirm

DIM  OurNewScreen%(15)   ' Laenge "NewScreen"-Datenstruktur = 32 Bytes


'---------------  "NewScreen"-Datenstruktur initialisieren -----------------'

OurNewScreenAdr& = VARPTR(OurNewScreen%(0))  ' Basisadresse "NewScreen"-Struktur
POKEW  OurNewScreenAdr& +  0, 0                 ' "LeftEdge"
POKEW  OurNewScreenAdr& +  2, 0                 ' "TopEdge"
POKEW  OurNewScreenAdr& +  4, 320               ' "Width"
POKEW  OurNewScreenAdr& +  6, 256               ' "Height"
```

```
POKEW  OurNewScreenAdr& +  8, 6                 ' "Depth"
POKE   OurNewScreenAdr& + 10, 0                 ' "DetailPen"
POKE   OurNewScreenAdr& + 11, 1                 ' "BlockPen"
POKEW  OurNewScreenAdr& + 12, EXTRAHALFBRITE%   ' "ViewModes"
POKEW  OurNewScreenAdr& + 14, CUSTOMSCREEN%     ' "Type"
POKEL  OurNewScreenAdr& + 16, 0                 ' "Font"
POKEL  OurNewScreenAdr& + 20, SADD(ScrTitle$)   ' "DefaultTitle"
POKEL  OurNewScreenAdr& + 24, 0                 ' "Gadgets"
POKEW  OurNewScreenAdr& + 28, 0                 ' "CustomBitMap"


OurScreen& = OpenScreen&(VARPTR(OurNewScreen%(0)))   ' Screen oeffnen !

IF OurScreen&=0 THEN END  ' Fehler beim Oeffnen des Bildschirms ?

ScrRPort& = OurScreen& + 84   ' Adresse des Screen-Rastports bestimmen


'----------------------------------------------------------------------------'


WinTitle$ = "Die Farben 0 - 31 im Fenster" + CHR$(0)

CLOSEWINDOW&    = 512    ' IDCMP-Flagge fuer Meldungen des Close-Gadget Klickens
WINDOWCLOSE&    =   8    ' Fenster mit Close-Gadget ausstatten
SIMPLEREFRESH& =  64    ' Simple-Refresh Methode fuer das Fenster

DIM  OurNewWindow&(12)  ' Laenge "NewWindow"-Datenstruktur = 48 Bytes


'----------------  "NewScreen"-Datenstruktur initialisieren ------------------'

OurNewWindowAdr& = VARPTR(OurNewWindow&(0))   ' Basisadresse "NewWindow"

POKEW  OurNewWindowAdr& +  0, 0                            ' "LeftEdge"
POKEW  OurNewWindowAdr& +  2, 126                          ' "TopEdge"
POKEW  OurNewWindowAdr& +  4, 320                          ' "Width"
POKEW  OurNewWindowAdr& +  6, 130                          ' "Height"
POKE   OurNewWindowAdr& +  8, 0                            ' "DetailPen"
POKE   OurNewWindowAdr& +  9, 1                            ' "BlockPen"
POKEL  OurNewWindowAdr& + 10, CLOSEWINDOW&                 ' "IDCMPFlags"
POKEL  OurNewWindowAdr& + 14, WINDOWCLOSE&+SIMPLEREFRESH&  ' "Flags"
POKEL  OurNewWindowAdr& + 18, 0                            ' "FirstGadget"
POKEL  OurNewWindowAdr& + 22, 0                            ' "CheckMark"
POKEL  OurNewWindowAdr& + 26, SADD(WinTitle$)              ' "Title"
POKEL  OurNewWindowAdr& + 30, OurScreen&                   ' "Screen"
POKEL  OurNewWindowAdr& + 34, 0                            ' "BitMap"
POKEW  OurNewWindowAdr& + 38, 0                            ' "MinWidth"
POKEW  OurNewWindowAdr& + 40, 0                            ' "MinHeight"
POKEW  OurNewWindowAdr& + 42, 320                          ' "MaxWidth"
POKEW  OurNewWindowAdr& + 44, 256                          ' "MaxHeight"POKEW
```

2

```
OurNewWindowAdr& + 46, CUSTOMSCREEN%                     ' "Type"


OurWindow& = OpenWindow&(VARPTR(OurNewWindow&(0)))   ' Fenster oeffnen !

IF OurWindow&=0 THEN                          ' Fehler beim Oeffnen des Fensters ?
 CALL CloseScreen(OurScreen&) : END   ' ja -> Bildschirm wieder schliessen
END IF

WinRPort& = PEEKL(OurWindow&+50)    ' Adresse des Window-Rastports bestimmen


'---------------------------------------------------------------------------------'


farbe% = 0                  ' Initialisierung der Variablen
x1% =  5   :  y1% = 15
x2% = 55   :  y2% = 55

WHILE farbe%<32                          ' Schleife für Rechtecke
 farbe% = farbe% + 1                     ' Farbregisternummer erhoehen
 x1% = x1% + 7                           ' neue obere linke Ecke und
 y1% = y1% + 1                           ' neue untere rechte Ecke des
 x2% = x2% + 8                           ' aktuellen Rechtecks berechnen
 y2% = y2% + 2
 CALL SetAPen(WinRPort&,farbe%)              ' Window-Zeichenfarbe setzen
 CALL RectFill(WinRPort&,x1%,y1%,x2%,y2%)   ' Rechteck im Window zeichnen
 CALL SetAPen(ScrRPort&,farbe%+32)           ' Screen-Zeichenfarbe setzen
 CALL RectFill(ScrRPort&,x1%,y1%,x2%,y2%)   ' Rechteck im Screen zeichnen
WEND

OurWinPort& = PEEKL(OurWindow&+86)    ' Zeiger auf Userport bestimmen

Msg& = WaitPort&(OurWinPort&)         ' auf Klicken des Close-Gadgets warten

CALL CloseWindow(OurWindow&)          ' Fenster schliessen
CALL CloseScreen(OurScreen&)          ' Bildschirm schliessen


END
```

3 DIE EXEC-LIBRARY

Die Exec-Library ist die »Basis-Library« des Amiga-Betriebssystems. Sie ist für das Multitasking und die Verwaltung anderer Libraries, Devices und Ressourcen zuständig. Eine Besonderheit dieser Library ist, daß sie immer geöffnet ist (man kann also Exec-Funktionen benutzen, ohne vorher die Library explizit öffnen zu müssen).

3.1 Wichtige Flags der Exec-Library

Die Memory-Flags

Diese Flags bestimmen die Eigenschaften eines Speicherbereiches. Sie werden an alle Routinen zur Speicherallokierung übergeben und werden in allen Datenstrukturen benutzt, die Speicherbereiche beschreiben (in Modula-2 sind sie in dem Aufzählungstyp *MemReqs* definiert, benutzt werden sie dann als *MemReqSet*).

C	Modula-2	Dez	Beschreibung
MEMF_PUBLIC	public	1	Der Speicher kann von anderen Tasks benutzt werden. In der aktuellen Version jedoch bedeutungslos.
MEMF_CHIP	chip	2	Der Speicherbereich soll im Chip-RAM liegen.
MEMF_FAST	fast	4	Der Speicherbereich soll im Fast-RAM liegen.
MEMF_CLEAR	memClear	65536	Der Speicherbereich soll beim Allokieren gelöscht (mit Nullen gefüllt) werden.
MEMF_LARGEST	largest	131072	Es soll der größtmögliche zusammenhängende Speicherblock der angegebenen Art alloziert werden.

3.2 Die Datenstrukturen der Exec-Library

Device

3

Offset		C (exec/devices.h)	Modula-2 (Exec)
Hex	Dez	struct Device	Device =
		{	RECORD
0000	0	struct Library dd_Library;	library:Library;
0012	34	};	END;

Assembler Verwenden Sie die identische *Library*-Datenstruktur.

Datenstrukturenreferenz
»Clipboard« : *IOClipboard.*
»Exec« : *IORequest, IOStdReq.*
»Printer« : *IOPrinter, IODRPReq.*

Routinenreferenz
»Exec« : *AddDevice, RemDevice.*

Beschreibung

Diese Prozedur beschreibt ein Device. Wie der Definition zu entnehmen ist, ist sie mit einer *Library*-Datenstruktur identisch.

dd_Library 0x0000 (0)

Eine normale *Library*-Datenstruktur, die das Device beschreibt.

ExecBase

Offset		C (exec/execbase.h)	Modula-2 (Exec)
Hex	Dez	struct ExecBase	ExecBase =
		{	RECORD
0000	0	struct Library LibNode;	libNode:Library;
0022	34	UWORD SoftVer;	softVer:CARDINAL;
0024	36	WORD LowMemChkSum;	lowMemChkSum:INTEGER;
0026	38	ULONG ChkBase;	chkBase:LONGCARD;
002A	42	APTR ColdCapture;	coldCapture:ADDRESS;
002E	46	APTR CoolCapture;	coolCapture:ADDRESS;
0032	50	APTR WarmCapture;	warmCapture:ADDRESS;
0036	54	APTR SysStkUpper;	sysStkUpper:ADDRESS;
003A	58	APTR SysStkLower;	sysStkLower:ADDRESS;

Offset		C (exec/execbase.h)	Modula-2 (Exec)
Hex	Dez		
003E	62	ULONG MaxLocMem;	maxLocMem:LONGCARD;
0042	66	APTR DebugEntry;	debugEntry:ADDRESS;
0046	70	APTR DebugData;	debugData:ADDRESS;
004A	74	APTR AlertData;	alertData:ADDRESS;
004E	78	APTR MaxExtMem;	maxExtMem:ADDRESS;
0052	82	UWORD ChkSum;	chkSum:CARDINAL;
0054	84	struct IntVector IntVects[16];	intVects:IntFlags IntVector;
0114	276	struct Task *ThisTask;	thisTask:TaskPtr;
0118	280	ULONG IdleCount;	idleCount:LONGCARD;
011C	284	ULONG DispCount;	dispCount:LONGCARD;
0120	288	UWORD Quantum;	quantum:CARDINAL;
0122	290	UWORD Elapsed;	elapsed:CARDINAL;
0124	292	UWORD SysFlags;	sysFlags:CARDINAL;
0126	294	BYTE IDNestCnt;	idNestCnt:Byte;
0127	295	BYTE TDNestCnt;	tdNestCnt:Byte;
0128	296	UWORD AttnFlags;	attnFlags:AttnFlagSet;
012A	298	UWORD AttnResched;	attnResched:CARDINAL;
012C	300	APTR ResModules;	resModules:ADDRESS;
0130	304	APTR TaskTrapCode;	taskTrapCode:PROC;
0134	308	APTR TaskExceptCode;	taskExceptCode:PROC;
0138	312	APTR TaskExitCode;	taskExitCode:PROC;
013C	316	ULONG TaskSigAlloc;	taskSigAlloc:LONGSET;
0140	320	UWORD TaskTrapAlloc;	taskTrapAlloc:BITSET;
0142	322	struct List MemList;	memList:List;
0150	336	struct List ResourceList;	resourceList:List;
015E	350	struct List DeviceList;	deviceList:List;
016C	364	struct List IntrList;	intrList:List;
017A	378	struct List LibList;	libList:List;
0188	392	struct List PortList;	portList:List;
0196	406	struct List TaskReady;	taskReady:List;
01A4	420	struct List TaskWait;	taskWait:List;
01B2	434	struct SoftIntList SoftInts[5];	softInts:ARRAY [0..4] OF SoftIntList;
0202	514	LONG LastAlert[4];	lastAlert:ARRAY [0..3] OF LONGINT;
0212	530	UBYTE VBlankFrequency;	vBlankFrequency:UByte;
0213	531	UBYTE PowerSupplyFrequency;	powerSupplyFrequency:UByte;
0214	532	struct List SemaphoreList;	semaphoreList:List;

Offset		C (exec/execbase.h)	Modula-2 (Exec)
Hex	Dez		
0222	546	APTR KickMemPtr;	kickMemPtr:ADDRESS;
0226	550	APTR KickTagPtr;	kickTagPtr:ADDRESS;
022A	554	APTR KickCheckSum;	kickCheckSum:LONGCARD;
022C	558	UBYTE ExecBaseReserved[10];	execBaseReserved:ARRAY [0..9] OF BYTE;
0238	568	UBYTE ExecBaseNewReserved[20];	execBaseNewReserved: ARRAY [0..19] OF BYTE;
024C	588	};	END;

Assembler — Syntax: Der erste Parameter (*LibNode*) ist nicht definiert.

Datenstrukturenreferenz — »PrtBase«: *PrinterData.*

Routinenreferenz — »Exec«: *FindTask.*
»ExecSupport«: *CreateTask*, *DeleteTask.*

Beschreibung

Die globale Basis-Datenstruktur von »Exec«, die die wichtigsten Sprungvektoren und Zustandsvariablen beinhaltet. Unüberlegte Veränderungen in dieser Datenstruktur haben immer nur ein Ziel: die Zerstörung aller im RAM befindlichen Daten.

- **LibNode 0x0000 (0)** — Eine eingebundene *Library*-Datenstruktur, die die Exec-Library beschreibt.
- **SoftVer 0x0022 (34)** — Die Versionsnummer des Kickstarts.
- **LowMemChkSum 0x0024 (36)** — Zur Zeit noch unbenutzt.
- **ChkBase 0x0026 (38)** — Das Komplement der Adresse der *ExecBase*-Datenstruktur. Es wird zur Prüfung der Position der *ExecBase* beim Reset benutzt und zur Position addiert.
- **ColdCapture 0x002A (42)** — Hier kann die Adresse einer Routine, die am Anfang eines Resets ausgeführt wird, eingetragen werden. Beim Einsprung beinhaltet das Adreßregister A5 die Rücksprungadresse. Dieser Vektor wird von der Reset-Routine automatisch auf *NULL* gesetzt.
- **CoolCapture 0x002E (46)** — Wie *ColdCapture*, wird allerdings später aufgerufen. Dieser Adreßzeiger wird nicht von der Reset-Routine gelöscht.
- **WarmCapture 0x0032 (50)** — Noch nicht benutzt.
- **SysStkUpper 0x0036 (54)** — Die untere Adresse des Systemstacks.

- **SysStkLower 0x003A (58)** Die obere Adresse des Systemstacks.
- **MaxLocMem 0x003E (62)** Die obere Grenze des Chip-RAM.
- **DebugEntry 0x0042 (66)** Die Einsprungsadresse des Debuggers. Sie können hier die Adresse Ihrer eigenen Debug-Prozedur oder des Debuggers eintragen.
- **DebugData 0x0046 (70)** Interne Daten für den Debugger.
- **AlertData 0x004A (74)** Interne Daten die für Alerts benutzt werden.
- **MaxExtMem 0x004E (78)** Die Adresse der oberen Grenze der Speichererweiterung oder *NULL* falls keine vorhanden ist.
- **ChkSum 0x0052 (82)** Eine Prüfsumme über die Felder *SoftVer* bis *MaxExtMem* der *ExecBase*-Datenstruktur. Bei Veränderung eines dieser Felder muß diese Summe neu berechnet werden.
- **IntVects 0x0054 (84)** Ein Array von IntVector-Datenstrukturen, die die Handler für die 15 möglichen Prozessorinterrupts beschreiben. Die einzelnen Felder stehen für folgende Interrupts:
- **ThisTask 0x0114 (276)** Ein Adreßzeiger auf die *Task*-Datenstruktur des momentan aktiven Tasks.
- **IdleCount 0x0118 (280)** Ein interner Zähler.
- **DispCount 0x011C (284)** Ein Zähler der nach jeder Taskumschaltung inkrementiert wird.

Nummer	Interrupt
0	Serieller Interrupt
1	Ende der Übertragung eines Diskblocks
2	Soft-Interrupt
3	Interrupt am CIA A
4	Copper-Interrupt
5	Vertikal Blank Interrupt. Wird ausgelöst, wenn der Rasterstrahl die Zeile 0 passiert. Über diesen Interrupt wird auch das Task-Switching gesteuert.
6	Wird vom Blitter nach Beendigung eines Jobs ausgelöst.
7	Wird vom Audiokanal 0 ausgelöst.
8	Wird vom Audiokanal 1 ausgelöst.
9	Wird vom Audiokanal 2 ausgelöst.
10	Wird vom Audiokanal 3 ausgelöst.
11	Wird durch serielle Eingabe ausgelöst.
12	Zur Synchronisation der Diskette.
13	Interrupt am CIA B.
14	Kann softwaremäßig ausgelöst werden.
15	NMI

- **Quantum 0x0120 (288)** Die Anzahl von »Zeitquanten«, die dem aktiven Task zur Verfügung stehen.
- **Elapsed 0x0122 (290)** Die Anzahl der seit dem Einschalten des Task bereits abgelaufenen Zeitquanten.
- **SysFlags 0x0124 (292)** Diverse Systemflags.
- **IDNestCnt 0x0126 (294)** Jedesmal, wenn *Disable* aufgerufen wird, wird dieser Zähler inkrementiert, wenn *Enable* aufgerufen wird, wird er dekrementiert. Ein Interrupt wird nur dann zugelassen, wenn dieser Zähler –1 ist.
- **TDNestCnt 0x0127 (295)** Jedesmal, wenn *Forbid* aufgerufen wird, wird dieser Zähler inkrementiert, wenn *Permit* aufgerufen wird, wird er dekrementiert. »Taskswitching« findet nur dann statt, wenn dieser Zähler –1 ist.
- **AttnFlags 0x0128 (296)** Gibt an, welche (Co-)Prozessoren verfügbar sind.

C	Modula-2	Dez	Beschreibung
AFF_68010	m68010	0	Der Motorola MC68010.
AFF_68020	m68020	1	Der Motorola MC68020.
AFF_68881	m68881	4	Der Coprozessor MC68881.

- **AttnResched 0x012A (298)** Wörtliche Übersetzung: »Aufmerksamkeit verschoben«.
- **ResModules 0x012C (300)** Ein Adreßzeiger auf eine Liste der residenten Module. Sie werden beim Reset aufgerufen.
- **TaskTrapCode 0x0130 (304)** Ein Adreßzeiger auf den Code, der aufgerufen wird, wenn ein Prozessortrap auftritt. Diese Adresse wird von dem gerade aktiven Task dann übernommen, wenn das Task keinen eigenen Trapcode hat.
- **TaskExceptCode 0x0134 (308)** Ein Adreßzeiger auf den Code, der aufgerufen wird, wenn in einem Task eine Exception auftritt. Diese Routine wird von dem gerade aktiven Task nur dann übernommen, falls sie keinen Exceptioncode hat.
- **TaskExitCode 0x0138 (312)** Ein Adreßzeiger auf den Code, der aufgerufen wird, wenn der Task den Prozessor verliert (durch das Multitasking ausgeschaltet wird). Diese Adresse wird von dem gerade aktiven Task dann übernommen, wenn der Task keinen eigenen Exitcode hat.
- **TaskSigAlloc 0x013C (316)** Die allokierten Signale des aktuellen Tasks
- **TaskTrapAlloc 0x0140 (320)** Die allokierten Traps des aktuellen Tasks

- **MemList 0x0142 (322)** Eine Liste (erzeugt durch eine eingebundene *List*-Datenstruktur), in der die freien Speicherbereiche verzeichnet sind.
- **ResourceList 0x0150 (336)** Eine Liste (erzeugt durch eine eingebundene *List*-Datenstruktur) aller *Resource*-Datenstrukturen, durch *List*-Datenstrukturen verkettet.
- **DeviceList 0x015E (350)** Eine Liste (erzeugt durch eine eingebundene *List*-Datenstruktur) der *DeviceList*-Datenstrukturen aller offenen Devices.
- **IntrList 0x016C (364)** Durch eine eingebundene *List*-Datenstruktur erzeugte Liste aller Interrupts.
- **LibList 0x017A (378)** Durch eine eingebundene *List*-Datenstruktur erzeugte Liste der *Library*-Datenstrukturen aller geöffneten Libraries.
- **PortList 0x0188 (392)** Durch eine eingebundene *List*-Datenstruktur erzeugte Liste aller Messageports.
- **LibList 0x0196 (406)TaskReady** Eine Liste (erzeugt durch eine eingebundene *List*-Datenstruktur) aller Tasks, die im »Ready«-Zustand sind, also auf ihren Anteil der Prozessorzeit warten.
- **TaskWait 0x01A4 (420)** Eine Liste (erzeugt durch eine eingebundene *List*-Datenstruktur) aller Tasks, die im »Wait«-Zustand sind, also keinen Anteil der Prozessorzeit beanspruchen, weil sie auf ein Signal warten.
- **SoftInts 0x01B2 (434)** In diesem Feld (erzeugt durch eine eingebundene *List*-Datenstruktur) stehen Listen von *Interrupt*-Datenstrukturen der Soft-Interrupts nach Priorität geordnet (–32 im Feld 0, –16 im Feld 1, 0 im Feld 2, 16 im Feld 3, 32 im Feld 3).
- **LastAlert 0x0202 (514)** Ein Adreßzeiger auf einen Zwischenspeicher für Alert-Daten, die von der Reset-Routine geholt werden.
- **VBlankFrequency 0x0212 (530)** Die Bildwiederholfrequenz (in der PAL-Version 50Hz).
- **VBlankFrequency 0x0213 (531) PowerSupplyFrequency** Die Frequenz der Netzspannung (in der PAL-Version 50 Hz).
- **SemaphoreList 0x0214 (532)** Durch eine eingebundene *List*-Datenstruktur erzeugte Liste aller *Semaphore*-Datenstrukturen.
- **KickMemPtr 0x0222 (546)** Ein Adreßzeiger auf eine *MemList*-Datenstruktur, die den Speicher beschreibt, der beim Reset für Systemzwecke allokiert wird.
- **KickTagPtr 0x0226 (550)** Ein Adreßzeiger auf eine *Resident*-Tabelle.
- **KickCheckSum 0x022A (554)** Die Prüfsumme des Kickstarts.
- **ExecBaseReserved 0x022C (558)** Ein privater Speicherplatz für »Exec«.

3

❑ **ExecBaseNewReserved**
0x0238 (568) — Noch ein privater Speicherplatz für »Exec«.

Interrupt

Offset		C (exec/interrupts.h)	Modula-2 (Exec)
Hex	Dez	struct Interrupt {	Interrupt = RECORD
0000	0	struct Node is_Node;	node:Node;
000E	14	APTR is_Data;	data:ADDRESS;
0012	18	VOID (*is_Code)();	code:PROC;
0016	22	};	END;

Assembler — Syntax: Strukturname = *IS*.
Definition der Parameter in Versalien.

Datenstrukturenreferenz — »Exec«: *ExecBase*.
»Intuition«: *IntuitionBase*.
»Resources«: *DiskResource*, *DiskResourceUnit*.

Routinenreferenz — »Exec«: *AddIntServer*, *Cause*, *RemIntServer*, *SetIntVector*.
»Resources«: *AddICRVector*, *RemICRVector*.

Beschreibung

Diese Datenstruktur wird zur Definition eines Interrupt-Handlers bzw. -Servers benutzt. Für jeden der 16 Hardware-Interrupts kann es nur einen Handler geben. Die *Interrupt*-Datenstrukturen der Default-Handler finden Sie in der *ExecBase-Datenstruktur*.

❑ **is_Node 0x0000 (0)** — Eine *Node*-Datenstruktur, die zur Einbindung des Servers in eine Liste dient. *Pri* enthält die Priorität des Servers.

❑ **is_Data 0x000E (14)** — Ein Adreßzeiger auf privaten Datenbereich der Interrupt-Routine. Wird beim Interruptaufruf in dem Adreßregister A1 an die Routine übergeben.

❑ **is_Code 0x0012 (18)** — Ein Adreßzeiger auf den auszuführenden Code. Beim Handler muß dieser mit *RTE*, beim Server mit *RTS* enden.

IntVector

Assembler — Syntax: Strukturname = *IV*.
Definition der Parameter in Versalien.

Datenstrukturenreferenz — Keine.

Routinenreferenz — Keine.

Offset		C (exec/interrupts.h)	Modula-2 (Exec)
Hex	Dez	struct IntVector	IntVector =
		{	RECORD
0000	0	APTR iv_Data;	data:ADDRESS;
0004	4	VOID (*iv_Code)();	code:PROC;
0008	8	struct Node *iv_Node;	node:NodePtr;
000C	12	};	END;

3

Beschreibung

Diese Datenstruktur wird Exec-intern benutzt, um einen Interrupt-Handler zu definieren. Dieser wird beim Auftreten eines Interrupts zuerst aufgerufen. Er sichert die Register und ruft nacheinander die Server auf.

- **iv_Data 0x0000 (0)** — Ein Adreßzeiger auf einen privaten Datenbereich der Behandlungsroutine. Dieser Adreßzeiger wird beim Auftreten eines Interrupts im Adreßregister A1 übergeben.
- **iv_Code 0x0004 (4)** — Ein Adreßzeiger auf die Behandlungsroutine.
- ***iv*_Node 0x0008 (8)** — Ein Adreßzeiger auf die *Node*-Datenstruktur des ersten Elementes der Liste der Server.

IORequest

Offset		C (exec/io.h)	Modula-2 (Exec)
Hex	Dez	struct IORequest	IORequest =
		{	RECORD
0000	0	struct Message io_Message;	message:Message;
0014	20	struct Device *io_Device;	device:DevicePtr;
0018	24	struct Unit *io_Unit;	unit:UnitPtr;
001C	28	UWORD io_Command;	command:CARDINAL;
001E	30	UBYTE io_Flags;	flags:IOFlagSet;
001F	31	BYTE io_Error;	error:Byte;
0020	32	};	END;

Assembler — Syntax: Strukturname = *IO*
Definition der Parameter in Versalien.

Datenstrukturenreferenz — »Audio«: *IOAudio*.
»Exec«: *IOStdReq* (indirekt).
»Timer«: *IOTimer*.

Routinenreferenz — »Exec«: *AbortIO*, *CheckIO*, *CloseDevice*, *DoIO*, *OpenDevice*, *SendIO*.

Beschreibung

Diese Datenstruktur dient zur Befehls- und Informationsübergabe an und von Devices. Die meisten Devices verwenden zur Kommunikation eine erweiterte Version, die *IOStdReq*-Datenstruktur, die ihrererseits noch zusätzlich um weitere Device-spezifische Felder erweitert ist. Eine solche Datenstruktur wird beim Öffnen eines Device initialisiert.

- **io_Message 0x0000 (0)** Eine eingebundene *Message*-Datenstruktur, die von den Devices benötigt wird, um nach der Beendigung der Befehlsausführung eine Antwort zu schicken.

- **io_Device 0x0014 (20)** Ein Adreßzeiger auf eine *Device*-Datenstruktur, die das Device beschreibt. Das Feld wird beim Öffnen der Device von »Exec« initialisiert.

- **io_Unit 0x0018 (24)** Dies ist ein Device-spezifisches Feld, das einen Adreßzeiger auf ein Unit enthält. Das wird normalerweise eine allgemeine »Exec«-*Unit*-Datenstruktur sein. Bei der Console-Device steht hier aber beispielsweise die Adresse einer *ConUnit*-Datenstruktur.

- **io_Command 0x001C (28)** Hier steht der Befehl, der an das Device geschickt wird. Neben den Device-spezifischen Befehlen, die in den entsprechenden Kapiteln behandelt werden, gibt es folgende Standardbefehle, die als Konstanten definiert sind.

C	Modula-2	Dez	Beschreibung
CMD_INVALID	invalid	0	Ein Befehl, der von dem Device nicht beantwortet werden muß.
CMD_RESET	reset	1	Einen Reset am Device durchführen, es also mit den Defaultwerten initialisieren.
CMD_READ	read	2	Veranlaßt das Device, eine Anzahl von Bytes in den Datenpuffer zu lesen.
CMD_WRITE	write	3	Veranlaßt das Device, eine Anzahl von Bytes aus dem Datenpuffer zu schreiben.
CMD_UPDATE	update	4	Veranlaßt das Device, den Inhalt aller internen Puffer auf das physikalische Gerät zu schreiben.
CMD_CLEAR	clear	5	Löscht alle internen Puffer des Device.
CMD_STOP	stop	6	Unterbricht alle Aktionen des Device.
CMD_START	start	7	Setzt die durch CMD_STOP unterbrochene Aktion fort.
CMD_FLUSH	flush	8	Veranlaßt das Device, alle IO/Requests zu unterbrechen und Fehlermeldungen zurückzugeben.
CMD_NONSTD	nonstd	9	Alle Device-spezifischen Befehle sind als CMD_NONSTD + x (mit x >= 0) definiert.

❏ **io_Flags 0x001E (30) 5**

Einige Flags, die den Status des Device bzw. befehlsspezifische Optionen bestimmt. Die unteren vier Bit sind für »Exec« reserviert, während die oberen Device-spezifisch sind. Der einzige vordefinierte Wert ist:

C	Modula-2	Dez	Beschreibung
IO_QUICK	quick	0	Den Request asynchron ausführen, also nicht auf die Erfolgs- oder Fehlermeldung warten.

❏ **io_Error 0x001F (31)**

Hier steht nach der Befehlsausführung die Fehlermeldung als Long-Zahl. Die Bedeutung der Meldung ist von Device zu Device verschieden. Sie finden die Device-spezifischen Fehlerwerte in den entsprechenden Abschnitten.

IOStdReq

Offset Hex	Offset Dez	C (exec/io.h)	Modula-2 (Exec)
		struct IOStdReq	IOStdReq =
		{	RECORD
0000	0	struct Message io_Message;	message:Message;
0014	20	struct Device *io_Device;	device:DevicePtr;
0018	24	struct Unit *io_Unit;	unit:UnitPtr;
001C	28	UWORD io_Command;	command:CARDINAL;
001E	30	UBYTE io_Flags;	flags:IOFlagSet;
001F	31	BYTE io_Error;	error:Byte;
0020	32	ULONG io_Actual;	actual:LONGCARD;
0024	36	ULONG io_Length;	length:LONGCARD;
0028	40	APTR io_Data;	data:ADDRESS;
002C	44	ULONG io_Offset;	offset:LONGCARD;
0030	48	};	END;

Assembler

Syntax: Strukturname = *IO*
Definition der Parameter in Versalien, die Größe der Datenstruktur steht in *IOSTD_SIZE*.

Datenstrukturenreferenz

»Audio«: *IOAudio*.
»Intuition«: *IntuitionBase*.
»Narrator«: *IONarrator*.
»Parallel«: *IOParallel*.
»Serial«: *IOSerial*.
»Trackdisk«: *IOTrackDisk*.

Routinenreferenz

»ExecSupport«: *CreateStdIO*, *DeleteStdIO*.

3

Beschreibung

Dies ist eine Erweiterung der oben beschriebenen *IORequest*-Datenstruktur. Die ersten sechs Felder von *IOStdReq* bilden eine *IORequest*-Datenstruktur. Eine um weitere Device-spezifische Felder erweiterte *IOStdReq*-Datenstruktur wird zum Öffnen und Betreiben der meisten Devices benötigt.

- **io_Message 0x0000 (0)** Eine eingebundene *Message*-Datenstruktur, die von den Devices benötigt wird, um nach der Beendigung der Befehlsausführung eine Antwort zu schicken.

- **io_Device 0x0014 (20)** Ein Adreßzeiger auf eine *Device*-Datenstruktur, die das Device beschreibt. Das Feld wird beim Öffnen der Devices von Exec initialisiert.

- **io_Unit 0x0018 (24)** Dies ist ein *Device*-spezifisches Feld, das einen Adreßzeiger auf ein Unit enthält. Das wird normalerweise eine allgemeine »Exec«-*Unit*-Datenstruktur sein. Bei dem Console-Device steht hier aber z.B. die Adresse einer *ConUnit*-Datenstruktur.

- **io_Command 0x001C (28)** Über dieses Feld stehen die aus der *IORequest*-Datenstruktur bekannten Kommandos zur Verfügung.

- **io_Flags 0x001E (30)** Diese Flags sind mit denen aus der *IORequest*-Datenstruktur identisch.

- **io_Error 0x001F (31)** Hier steht nach der Befehlsausführung die Fehlermeldung als Long-Zahl. Die Bedeutung der Meldung ist von Device zu Device verschieden. Sie finden die Device-spezifischen Fehlerwerte in den entsprechenden Abschnitten.

- **io_Actual 0x0020 (32)** Nach der Ausführung eines Befehls finden Sie hier die Anzahl der betroffenen (z.B. geschriebenen) Bytes.

- **io_Length 0x0024 (36)** Hier wird an das Device die Anzahl der Bytes, die von dem Befehl betroffen werden sollen, übergeben. Der Wert –1 bedeutet oft »alles« (was immer das auch im Einzelfall heißen mag).

- **io_Data 0x0028 (40)** Ein Adreßzeiger auf den Datenpuffer. Dies muß ein vom Benutzer allokierter Speicherbereich der richtigen Größe sein.

- **io_Offset 0x002C (44)** Gibt einen Device- und befehlsspezifischen Offset (z.B. Anfangsblock bei dem Trackdisk-Device) an.

Library

Offset		C (exec/libraries.h)	Modula-2 (Exec)
Hex	Dez	extern struct Library	Library =
		{	RECORD
0000	0	struct Node lib_Node;	node:Node;
000E	14	UBYTE lib_Flags;	flags:LibFlagSet;
000F	15	UBYTE lib_pad;	pad:BYTE;
0010	16	UWORD lib_NegSize;	negSize:CARDINAL;
0012	18	UWORD lib_PosSize;	posSize:CARDINAL;
0014	20	UWORD lib_Version;	version:CARDINAL;
0016	22	UWORD lib_Revision;	revision:CARDINAL;
0018	24	APTR lib_IdString;	idString:ADDRESS;
001C	28	ULONG lib_Sum;	sum:LONGCARD;
0020	32	UWORD lib_OpenCnt;	openCnt:CARDINAL;
0022	34	};	END;

Assembler

Syntax: Strukturname = *LIB*
Definition der Parameter in Versalien, wobei *lib_Node* nicht definiert ist.

Datenstrukturenreferenz

»Dos«: *DosLibrary.*
»Exec«: *Device, ExecBase.*
»Graphics« : *GfxBase.*
»Intuition«: *IntuitionBase.*
»PrtBase« : *DeviceData.*
»Resources«: *DiskResource, MiscResource.*

Routinenreferenz

»Exec«: *AddLibrary, CloseLibrary, MakeLibrary, OldOpenLibrary, OpenLibrary, RemLibrary, SetFunction, SumLibrary.*

Beschreibung

Diese Datenstruktur enthält alle Informationen, die »Exec« zur Verwaltung einer Library braucht. Sie wird beim Erstellen einer neuen Library mit der Routine *MakeLibrary* automatisch von »Exec« mit den richtigen Werten initialisiert.

- **lib_Node 0x0000 (0)** — Eine eingebundene *Node*-Datenstruktur, die zur Integrierung der Library in die Liste der Systemlibraries dient.

- **lib_Flags 0x000E (14)** — Diese Flags definieren den Zustand einer Library. Hier ist eine Kombination der folgenden Werte (Bits) möglich:

3

C	Modula-2	Dez	Beschreibung
LIBF_SUMMING	summing	1	Es gibt einen Task, der gerade die Checksumme berechnet.
LIBF_CHANGED	changed	2	Es wurden eine oder mehrere Funktionen der Library verändert.
LIBF_SUMUSED	sumUsed	4	Veranlaßt einen Reset, falls ein Prüfsummenfehler auftritt.
LIBF_DELEXP	delExp	8	Die Library soll geschlossen werden, sie wird aber von einem anderen Task benutzt. Das Schließen wird verschoben.

- **lib_Pad 0x000F (15)** Ein schnuckeliges Füllbyte.
- **lib_NegSize 0x0010 (16)** Dieses Wort gibt die Größe der Sprungtabelle für die Routinen der Library in Bytes an.
- **lib_PosSize 0x0012 (18)** Dieses Wort gibt die Größe der Daten in Bytes an.
- **lib_Version 0x0014 (20)** Hier steht die Versionsnummer der Library. Sie wird beim Öffnen überprüft.
- **lib_Revision 0x0016 (22)** Hier steht die Überarbeitungsnummer der Library.
- **lib_IdString 0x0018 (24)** Ein Adreßzeiger auf den Namen (als nullterminierte Zeichenkette) der Library. Dieser Name wird zum Öffnen benutzt (z.B. »dos.library«).
- **lib_Sum 0x001C (28)** Hier steht die Checksumme der Library, die mit der *SumLibrary*-Routine berechnet wird.
- **lib_OpenCnt 0x0020 (32)** Dieses Feld gibt an, wie viele Tasks diese Library für sich geöffnet hat.

List

Offset		C (exec/lists.h)	Modula-2 (Exec)
Hex	Dez	struct List	List =
		{	RECORD
0000	0	struct Node *lh_Head;	head:NodePtr;
0004	4	struct Node *lh_Tail;	tail:NodePtr;
0008	8	struct Node *lh_TailPred;	tailPred:NodePtr;
000C	12	UBYTE lh_Type;	type:NodeType;
000D	13	UBYTE lh_pad;	pad:BYTE;
000E	14	};	END;

Assembler

Syntax: Strukturname = *LH*
Definition der einzelnen Parameter in Versalien.

Datenstrukturenreferenz

»Exec«: *ExecBase, MsgPort, SoftIntList, Task.*
»Graphics« : *GfxBase, LayerInfo.*
»Intuition«: *IntuitionBase.*
»KeyMap«: *KeyMapResource.*
»Resources«: *DiskResource.*
»Workbench«: *FreeList.*

Routinenreferenz

»Exec«: *AddHead, AddTail, Enqueue, Insert, ObtainSemaphore, ReleaseSemaphoreList, RemHead.*

»ExecSupport«: *NewList.*

Beschreibung

Diese Datenstruktur dient als Header für Listen, die aus *Node*-Datenstrukturen aufgebaut sind. Die »Nodes« sind oft in andere Datenstrukturen eingebunden, die die eigentlichen Elemente der Liste sind. Sie dienen lediglich zur Verknüpfung der Elemente (siehe *Node*-Datenstruktur). Ein Adreßzeiger auf die *List*-Datenstruktur ist die Kennung einer Liste, die an diverse Listenroutinen übergeben wird.

- **lh_Head 0x0000 (0)** — Ein Adreßzeiger auf die erste *Node*-Datenstruktur der Liste.
- **lh_Tail 0x0004 (4)** — Dieses Feld ist immer NULL.
- **lh_TailPred 0x0008 (8)** — Ein Adreßzeiger auf die letzte *Node*-Datenstruktur der Liste.
- **lh_Type 0x000C (12)** — Dieser Wert enthält Informationen über die Elemente der Liste. Er sollte identisch mit den entsprechenden Werten der *Type*-Felder aus den *Node*-Datenstrukturen der Liste sein.

C	Modula-2	Dez	Beschreibung
NT_UNKNOWN	unknown	0	Die Liste enthält unbekannte Elemente.
NT_TASK	task	1	Die Liste enthält *Task*-Datenstrukturen.
NT_INTERRUPT	interrupt	2	Die Liste enthält Interrupt-bezogene Datenstrukturen.
NT_DEVICE	device	3	Die Liste enthält *DeviceList*-Datenstrukturen.
NT_MSGPORT	msgPort	4	Die Liste enthält *MsgPort*-Datenstrukturen.
NT_MESSAGE	message	5	Die Liste enthält Messages.
NT_FREEMSG	freeMsg	6	Die Liste enthält nur eine Message.
NT_REPLYMSG	replyMsg	7	Enthält eine Message, die gerade beantwortet wurde und an den Absender zurückgeht.
NT_RESOURCE	resource	8	Die Liste enthält Resources.
NT_LIBRARY	library	9	Die Liste enthält *Library*-Datenstrukturen.

C	Modula-2	Dez	Beschreibung
NT_MEMORY	memory	10	Dies ist eine Speicherliste.
NT_SOFTINT	softInt	11	Die Liste enthält die *IntVector*-Datenstrukturen der Soft-Interrupts.
NT_FONT	font	12	Die Liste enthält Font-Daten.
NT_PROCESS	process	13	Die Liste enthält *Process*-Datenstrukturen.
NT_SEMAPHORE	semaphore	14	Die Liste enthält *Semaphore*-Datenstrukturen.
NT_SIGNALSEM	signalSem	15	Liste enthält *SignalSemaphore*-Datenstrukturen.

Wenn Sie die *List*en bzw. die *Node*s für Ihre eigenen Zwecke benutzen, dann können Sie auf die Initialisierung dieses Feldes verzichten. Falls Sie Exec-Listen bearbeiten, kann eine fehlende Initalisierung manchmal zu Problemen führen.

❑ **lh_pad 0x000D (13)** Ein einsames Füllbyte.

MemChunk

Offset		C (exec/memory.h)	Modula-2 (Exec)
Hex	Dez	struct MemChunk	MemChunk =
		{	RECORD
0000	0	struct MemChunk *mc_Next;	next:MemChunkPtr;
0004	4	ULONG mc_Bytes;	bytes:LONGCARD;
0008	8	};	END;
		/* Freier Speicherblock */	

Assembler Syntax: Strukturname = *MC*.
Definition der Parameter in Versalien.

Datenstrukturenreferenz »Exec«: *MemHeader*.

Routinenreferenz Keine.

Beschreibung

Eine Reihe solcher Datenstrukturen bilden zusammen mit einer *MemHeader*-Datenstruktur, die als Kopf dient, eine Liste freier Speicherbereiche. Eine *MemChunk*-Datenstruktur steht am Anfang eines freien Speicherbereiches und beinhaltet einen Verkettungszeiger auf den nächsten Bereich sowie die Größe des Speicherbereichs.

❑ **mc_Next 0x0000 (0)** Hier steht ein Adreßzeiger auf die nächste *MemChunk*-Datenstruktur in der Liste oder *NULL*.

❑ **mc_Bytes 0x0004 (4)** Dieses Feld gibt die Länge des Speicherbereiches an, der direkt hinter diesem Langwort beginnt.

MemEntry

Offset		C (exec/memory.h)	Modula-2 (Exec)
Hex	**Dez**	struct MemEntry	MemEntry =
		{	RECORD
		union	CASE :INTEGER OF
0000	0	{	
		ULONG meu_Reqs;	| 1:reqs:MemReqSet
		APTR meu_Addr;	| 2:addr:ADDRESS
		}	
		me_Un;	END;
0004	4	ULONG me_Length;	length:LONGCARD;
0008	8	};	END;

Assembler

Syntax: Strukturname = *ME*
meu_Reqs —> *ME_REQS*
meu_Addr —> *ME_ADDR*
me_Length —> *ME_LENGTH*

Datenstrukturenreferenz »Exec«: *MemList.*

Routinenreferenz Keine.

Beschreibung

Als Element eines Arrays beschreibt diese Datenstruktur am Ende einer *MemList*-Datenstruktur einen der Speicherbereiche, die alloziert werden sollen (siehe *MemList*-Datenstruktur). Vor dem Aufruf von *AllocEntry* muß diese Datenstruktur die Anforderungen an den zu allozierenden Speicherbereich enthalten, nach einer erfolgreichen Allozierung finden Sie dort die Adresse des allozierten Speicherbereiches.

❑ **meu_Reqs 0x0000 (0)** Vor dem Aufruf von *AllocEntry* stehen hier die Flags, die die Anforderungen an den Speicher angeben (siehe »Die Memory-Flags«).

❑ **meu_Addr 0x0000 (0)** Hier steht nach einer erfolgreichen Allozierung die Adresse des allozierten Speicherbereiches.

❑ **me_Length 0x0004 (4)** In diesem Feld wird die Länge des zu allokierenden Speicherbereiches in Byte angegeben.

3

MemHeader

Offset		C (exec/memory.h)	Modula-2 (Exec)
Hex	Dez	struct MemHeader	MemHeader =
		{	RECORD
0000	0	struct Node mh_Node;	node:Node;
000C	14	UWORD mh_Attributes;	attributes:MemTypeSet;
0010	16	struct MemChunk *mh_First;	first:MemChunkPtr;
0014	20	APTR mh_Lower;	lower:ADDRESS;
0018	24	APTR mh_Upper;	upper:ADDRESS;
001C	28	ULONG mh_Free;	free:LONGCARD;
0020	32	};	END;

Assembler Syntax: Strukturname = *MH*
Definition der Parameter in Versalien, wobei *mh_Node* nicht definiert ist.

Datenstrukturenreferenz Keine.

Routinenreferenz »Exec« : *Allocate*, *Deallocate*.

Beschreibung

Mit Hilfe dieser Datenstruktur und den *Allocate*- und *Deallocate*-Routinen können Sie Ihren eigenen Speicherbereich, den Sie einmal mit *AllocMem* alloziert haben, auch selbst verwalten. Sie ist der Kopf einer Liste von zusammenhängenden, freien Speicherbereichen, von denen jeder von einer *MemChunk*-Datenstruktur eingeleitet wird. Mit *Allocate* und *Deallocate* können Sie Speicherbereiche aus Ihrer Liste allozieren und wieder freigeben.

- **mh_Node 0x0000 (0)** Eine eingebundene *Node*-Datenstruktur, mit deren Hilfe Sie mehrere Speicherlisten verketten können.

- **mh_Attributes 0x000C (14)** Einige Flags, die die Eigenschaften des Speicherbereiches beschreiben (siehe »Die MemoryFlags«).

- **mh_MemChunk 0x0010 (16)** Hier steht ein Adreßzeiger auf den ersten freien Speicherbereich (oder genauer auf die *MemChunk*-Datenstruktur, die ihn einleitet).

- **mh_Lower 0x0014 (20)** Dieses Feld beinhaltet die unterste Adresse des gesamten, in der Liste enthaltenen Speicherbereiches.

- **mh_Upper 0x0018 (24)** Die oberste Adresse des gesamten, in der Liste enthaltenen Speicherbereiches.

- **mh_Free 0x001C (28)** Die Größe des in der gesamten Liste enthaltenen freien Speicherbereiches (also die Summe über alle *MemChunks*) der Liste.

MemList

Offset		C (exec/memory.h)	Modula-2 (Exec)
Hex	Dez	struct MemList	MemList =
		{	RECORD
0000	0	struct Node ml_Node;	node:Node;
000E	14	UWORD ml_NumEntries;	numEntries:CARDINAL;
0010	16	struct MemEntry ml_ME[1];	(*me:ARRAY [0..munEntries-1] OF MemEntry;*)
0018	24	};	END;

Assembler — Syntax: Strukturname = *ML*
Definition der einzelnen Parameter in Versalien, wobei *ml_Node* nicht definiert ist.

Datenstrukturreferenz — Keine.

Routinenreferenz — »Exec«: *AllocEntry*, *FreeEntry*.

Beschreibung

Mit Hilfe des dieser Datenstruktur nachfolgenden Arrays von *MemEntry*-Datenstrukturen, kann man mit einem Aufruf von *AllocEntry* mehrere verschiedene Speicherbereiche allozieren und später wieder mit *FreeEntry* freigeben. Die Beschreibung der einzelnen Speicherbereiche steht in den *MemEntry*-Datenstrukturen (siehe *MemEntry*-Datenstruktur). Dort stehen nach einer erfolgreichen Allozierung auch die Adressen der allozierten Speicherblöcke.

❑ **ml_Node 0x0000 (0) u** — Dies ist eine eingebundene *Node*-Datenstruktur, mit deren Hilfe man mehrere *MemList*-Datenstrukturen verketten kann. Dieses Feld wird sowohl von *AllocEntry* als auch von *FreeEntry* ignoriert.

❑ **ml_NumEntries 0x000E (14)** — Hier steht die Anzahl der nachfolgenden *MemEntry*-Datenstrukturen.

❑ **ml_ME[1] 0x0010 (16)** — Das erste Element des Arrays von *MemEntry*-Datenstrukturen (die anderen folgen direkt danach).

3

Message

Offset		C (exec/ports.h)	Modula-2 (Exec)
Hex	Dez	struct Message	Message =
		{	RECORD
0000	0	struct Node mn_Node;	node:Node;
000E	14	struct MsgPort *mn_ReplyPort;	replyPort:MsgPortPtr;
0012	18	UWORD mn_Length;	length:CARDINAL;
0014	20	};	END;

Assembler Syntax: Strukturname = MN
Definition der einzelnen Parameter in Versalien, wobei *mn_Node* nicht extra definiert ist.

Datenstrukturenreferenz
»Audio«: *IOAudio.*
»Clipboard«: *IOClipboard, SatisfyMsg.*
»Dos«: *DosPacket, FileHandle, StandardPacket.*
»Exec«: *IORequest, IOStdReq.*
»Graphics«: *TextFont.*
»Intuition«: *IntuiMessage.*
»Printer« : *IODRPReq, IOPrinter.*
»Resources«: *DiskResourceUnit.*
»Workbench«: *WBStartup.*

Routinenreferenz »Exec«: *PutMsg, ReplyMsg.*

Beschreibung

Diese Datenstruktur dient zur Übermittlung von Nachrichten zwischen Tasks und Prozessen. Eine *Message*-Datenstruktur steht normalerweise am Anfang einer anderen Datenstruktur, die die eigentliche Nachricht darstellt. Manchmal wird auch die Adresse der eigentlichen Nachricht in das *Name*-Feld der *Node*-Datenstruktur eingetragen.

❑ **mn_Node 0x0000 (0)** Eine eingebundene *Node*-Datenstruktur, die das Verketten von mehreren *Message*-Datenstrukturen (z.B. in der Warteschleife eines *Messageports*) ermöglicht.

❑ **mn_ReplyPort 0x000E (14)** Die Adresse des *Messageports*, an den die Antwort von der Routine *ReplyMsg* geschickt wird.

❑ **mn_Length 0x0012 (18)** Hier wird die Länge der eigentlichen Nachricht in Bytes eingetragen.

3

MinList

Offset		C (exec/lists.h)	Modula-2 (Exec)
Hex	Dez	struct MinList	MinList =
		{	RECORD
0000	0	struct MinNode *mlh_Head;	head:MinNodePtr;
0004	4	struct MinNode *mlh_Tail;	tail:MinNodePtr;
0008	8	struct MinNode *mlh_TailPred;	tailPred:MinNodePtr;
000C	12	};	END;

Assembler — Syntax: Strukturname = *MLH*
Definition der einzelnen Parameter in Versalien.

Datenstrukturenreferenz — »Exec«: *SignalSemaphore*.
»Graphics«: *LayerInfo*.

Routinenreferenz — Keine.

Beschreibung

Die *MinList*-Datenstruktur ist eine gekürzte Version der *List*-Datenstruktur. Sie wird immer dann an Stelle der *List*-Datenstruktur als Listenheader benutzt, wenn keine Angabe über den Typ der Elemente notwendig ist. Die Elemente der Liste werden dann über *MinNode*-Datenstrukturen statt über *Node*-Datenstrukturen verkettet.

- **mlh_Head 0x0000 (0)** — Ein Adreßzeiger auf die erste *MinNode*-Datenstruktur der Liste.
- **mlh_Tail 0x0004 (4)** — Dieses Feld ist immer *NULL*.
- **mlh_TailPred 0x0008 (8)** — Ein Adreßzeiger auf die letzte *MinNode*-Datenstruktur der Liste.

MinNode

Offset		C (exec/nodes.h)	Modula-2 (Exec)
Hex	Dez	struct MinNode	MinNode =
		{	RECORD
0000	0	struct MinNode *mln_Succ;	succ:MinNodePtr;
0004	4	struct MinNode *mln_Pred;	pred:MinNodePtr;
0008	8	};	END;

Assembler — Syntax: Strukturname = *MLN*
Definition der einzelnen Parameter in Versalien.

Datenstrukturenreferenz — »Exec«: *MinList*, *SemaphoreRequest*.

Routinenreferenz — »Exec«: *AddTail*, *Enque*, *Insert*, *Remove*.

3

Beschreibung

Dies ist eine vereinfachte *Node*-Datenstruktur. Sie kann zusammen mit der *MinList*-Datenstruktur zur Bildung von Listen benutzt werden. *MinList* und *MinNode* können immer dann anstelle von *List* und *Node* benutzt werden, wenn die Typen-, Prioritäts- und Namensangaben nicht benötigt werden.

- **mln_Succ 0x0000 (0)** — Hier steht ein Adreßzeiger auf die *Node*-Datenstruktur des nächsten Elements der Liste.

- **mln_Pred 0x0004 (4)** — Hier steht ein Adreßzeiger auf die *Node*-Datenstruktur des Vorgängerelements.

MsgPort

Offset Hex	Offset Dez	C (exec/ports.h)	Modula-2 (Exec)
		struct MsgPort	MsgPort =
		{	RECORD
0000	0	struct Node mp_Node;	node:Node;
000E	14	UBYTE mp_Flags;	CASE flags:MsgPortAction OF
			| signal:
000F	15	UBYTE mp_SigBit;	sigBit:UByte;
0010	16	struct Task *mp_SigTask;	sigTask:TaskPtr;
			| softint:
			pad0:BYTE;
			softInt:InterruptPtr;
			| ignore:
			pad1:BYTE;
			pad2:ADDRESS
			END;
0014	20	struct Lis mp_MsgList;	msgList:List;
0022	34	};	END;

Assembler — Syntax: Strukturname = *MP*
Definition der einzelnen Parameter in Versalien, wobei *mp_Node* nicht extra definiert ist.

Datenstrukturenreferenz
»ConUnit«: *ConUnit*.
»Dos«: *DosPacket*, *FileHandle*, *Process*.
»Exec«: *ExecBase*, *Message*, *Semaphore*, *Unit*.
»Intuition«: *Preferences*, *Window*.

	»PrtBase«:	*PrinterData.*
	»Workbench«:	*WBStartup.*
Routinenreferenz	»Exec«:	*FindPort, GetMsg, Procure, PutMsg, RemPort, Vacate, WaitPort.*
	»ExecSupport«:	*CreatePort, DeletePort, CreateExtIO, CreateStdIO.*

Beschreibung

Diese Datenstruktur bildet die Grundlage für die Kommunikation zwischen verschiedenen Elementen des Multitasking. Sie definiert einen »Empfänger« (*Messageport*), an den Nachrichten in Form von *Message*-Datenstrukturen geschickt werden können. Ankommende *Message*s werden in einer Liste an den *Port* gehängt. Die erste Nachricht in der Liste kann mit der Routine *GetMsg* geholt und aus der Liste entfernt werden. Bei Ankunft einer Nachricht wird an den in *mp_SigTask* eingetragenen Task (falls er zu einem Task gehört) ein Signal gesendet. Dieser kann den Task zu verschiedenen Handlungen veranlassen. Eine Nachricht an einem *Port* kann auch einen Softwareinterrupt auslösen.

❑ **mp_Node 0x0000 (0)**

Eine eingebundene *Node*-Datenstruktur, die zur Verkettung mehrerer *MsgPort*s in einer Liste benutzt werden kann. Das *ln_Name*-Feld kann dazu benutzt werden, mit der Routine *FindPort* einen bestimmten Port zu finden.

❑ **mp_Flags 0x000E (14)**

Dieses Feld gibt an, zu welcher Handlung der in *mp_SigTask* eingetragene Task durch die Ankunft einer Nachricht an diesem *MsgPort* veranlaßt werden soll. Hier können folgende Werte stehen:

C	Modula-2	Dez	Beschreibung
PA_SIGNAL	signal	0	Bei Ankunft einer Nachricht soll ein Signal geschickt werden.
PA_SOFTINT	softint	1	Es soll bei Ankunft einer Nachricht ein Softwareinterrupt ausgelöst werden.
PA_IGNORE	ignore	2	Bei Ankunft einer Nachricht soll nichts geschehen.

❑ **mp_SigBit 0x000F (15)**

Gibt die Nummer des Signalbits an, das bei Ankunft einer Nachricht am Port an den Task geschickt wird, der durch *mp_SigTask* bestimmt ist. Dieser kann dann z.B. eine Exception auslösen, also den *Wait*-Zustand beenden.

❑ **mp_SigTask 0x0010 (16)**

Falls *mp_Flags* den Wert *PA_SIGNAL* hat, ist dies ein Adreßzeiger auf die *Task*-Datenstruktur des Tasks, für den die Nachrichten dieses Ports bestimmt sind. Sollte in *mp_Flags* aber *PA_SOFTINT* stehen, so befindet sich hier die Adresse einer entsprechenden *Interrupt*-Datenstruktur. Hier kann auch *NULL* stehen, dann muß aber *mp_Flags* den

Wert *PA_IGNORE* haben, damit nicht versucht wird, an einen nicht existierenden Task ein Signal zu senden.

❑ **mp_MsgList 0x0014 (20)**

Hier werden alle angekommenen Nachrichten in einer Liste von *Message*-Datenstrukturen gespeichert, bis sie durch *GetMsg* abgeholt werden.

3

Node

Offset		C (exec/nodes.h)	Modula-2 (Exec)
Hex	**Dez**	struct Node	Node =
		{	RECORD
0000	0	struct Node *ln_Succ;	succ:NodePtr;
0004	4	struct Node *ln_Pred;	pred:NodePtr;
0008	8	UBYTE ln_Type;	type:NodeType;
0009	9	BYTE ln_Pri;	pri:Byte;
000A	10	char *ln_Name;	name:ADDRESS;
000E	14	};	END;

Assembler

Syntax: Strukturname = *LN*
Definition der einzelnen Parameter in Versalien.

Datenstrukturenreferenz

»Clipboard«: *ClipboardUnitPartial.*
»DiskFont«: *DiskFontHeader.*
»Exec«: *IntVector, Interrupt, Library, List, MemHeader, MemList, Message, MsgPort, Task.*
»Expansion«: *ConfigDev.*
»Graphics«: *Isrvstr.*
»KeyMap«: *KeyMapNode, KeyMapResource.*

Routinenreferenz

»Exec«: *AddTail, Enque, Insert, Remove.*

Beschreibung

Diese Datenstruktur dient zur Verknüpfung der Elemente einer Liste. Die eigentlichen Elemente stehen normalerweise direkt hinter dem letzten Feld der *Node*-Datenstruktur. In die meisten Datenstrukturen, die zu Listen verknüpft werden, wird daher meistens als erstes Element ein *Node* eingebunden. Gelegentlich dient auch das *ln_Name*-Feld dazu, den *Node* mit dem Inhalt zu verbinden. Den Header einer Liste bildet immer eine *List*-Datenstruktur.

❑ **ln_Succ 0x0000 (0)**

Hier steht ein Adreßzeiger auf die *Node*-Datenstruktur des nächsten Elements der Liste.

❑ **ln_Pred 0x0004 (4)**

Hier steht ein Adreßzeiger auf die *Node*-Datenstruktur des Vorgängerelements.

❏ **ln_Type 0x0008 (8)**

In dieses Feld schreibt Exec den Typ des Elements. Die Werte sind mit den bei *List* beschriebenen identisch. Es ist nur bei Exec-internen Listen von Bedeutung.

❏ **ln_Pri 0x0009 (9)**

Dieses Feld gibt die Priorität an, die zwischen –128 und 127 liegen darf. Dieser Wert ist für einige Systemdatenstrukturen, die als erstes Feld eine *Node*-Datenstruktur besitzen, von Bedeutung. So wird z.B. die Priorität eines Tasks durch das *ln_Pri*-Feld der in die *Task*-Datenstruktur eingebundenen *Node*-Datenstruktur bestimmt.

❏ **ln_Name 0x000A (10)**

Hier kann ein Adreßzeiger auf eine Zeichenkette stehen, die den Namen des Elements (z.B. Taskname), enthält.

Resident

Offset		C (exec/resident.h)	Modula-2 (Exec)
Hex	**Dez**	struct Resident	Resident =
		{	RECORD
0000	0	UWORD rt_MatchWord;	matchWord:CARDINAL;
0002	2	struct Resident *rt_MatchTag;	matchTag:ResidentPtr;
0006	6	APTR rt_EndSkip;	endSkip:ADDRESS;
000A	10	UBYTE rt_Flags;	flags:ResidentFlagSet;
000B	11	UBYTE rt_Version;	version:UByte;
000C	12	UBYTE rt_Type;	type:NodeType;
000D	13	BYTE rt_Pri;	pri:Byte;
000E	14	char *rt_Name;	name:ADDRESS;
0012	18	char *rt_IdString;	idString:ADDRESS;
0016	22	APTR rt_Init;	init:ADDRESS;
001A	26	};	END;

Assembler

Syntax: Strukturname = *RT*
Definition der einzelnen Parameter in Versalien.

Datenstrukturenreferenz

»Exec«: *ExecBase*, *Resident*.

Routinenreferenz

»Exec«: *FindResident*, *InitResident*.

Beschreibung

Diese Datenstruktur beschreibt ein Reset-festes Modul. Nach einem Reset wird der Speicher nach solchen Datenstrukturen abgesucht. Sie werden in eine Liste in der *ExecBase* eingetragen und abgearbeitet. Dabei wird, je nach Inhalt der Datenstruktur, entweder ein Programm ausgeführt oder eine Library, Device oder Ressource erzeugt und in das System eingefügt. Mit Hilfe der *Resident*-Datenstruktur ist es möglich, Reset-feste Programme zu schreiben.

- **rt_MatchWord 0x0000 (0)** Hier muß der Wert 0x4AFC (19196) stehen, damit eine *Resident*-Datenstruktur als solche erkannt wird. Nach einem Reset wird nach diesem Wert gesucht, um die *Resident*-Datenstrukturen zu erkennen.

- **rt_MatchTag 0x0002 (2)** Ein Adreßzeiger zurück auf den Anfang der Datenstruktur. Dieser Adreßzeiger wird bei der Suche nach *Resident*-Datenstrukturen überprüft, wenn das richtige *MatchWord* gefunden wurde. Erst wenn der *MatchTag* stimmt, glaubt das System, daß es auch tatsächlich eine *Resident*-Datenstruktur gefunden hat.

- **rt_EndSkip 0x0006 (6)** Hier steht ein Adreßzeiger auf das Ende der Datenstruktur, inklusiv der eventuell dahinter stehenden Daten.

- **rt_Flags 0x000A (10)** Die hier stehenden Flags bestimmen, was bei der Abarbeitung der *Resident*-Datenstruktur passieren soll.

C	Modula-2	Dez	Bedeutung
RTF_COLDSTART	coldstart	1	Es soll der in *rt_Init* angegebene Befehl ausgeführt werden.
RTF_AUTOINIT	autoinit	128	Es soll der in *rt_Type* angegebene Befehl ausgeführt werden.

- **rt_Version 0x000B (11)** Die Version der Datenstruktur.

- **rt_Type 0x000C (12)** Dieses Feld enthält, falls das *RTF_AUTOINIT*-Flag in *rt_Flags* gesetzt ist, den auszuführenden Befehl. Es kann sich dabei um einen der drei folgenden Werte handeln:

Dez	Bedeutung
3	Eine Device mit *AddDevice* hinzufügen
8	Eine Resource mit *AddResource* hinzufügen
9	Eine Library mit *AddLibrary* hinzufügen

- **rt_Pri 0x000D (13)** Gibt die Priorität der *Resident*-Datenstruktur an.

- **rt_Name 0x000E (14)** Hier kann ein Adreßzeiger auf den Namen der *Resident*-Datenstruktur stehen.

- **rt_IdString 0x0012 (18)** Hier kann die Adresse eines Strings, der weitere Informationen zu dieser Datenstruktur beinhaltet, stehen.

- **rt_Init 0x0016 (22)** Entweder ein Adreßzeiger auf das auszuführende Programm oder, falls *MakeLib* aufgerufen wird, die Sprungtabelle der Library.

3

Semaphore

Offset		C (exec/semaphores.h)	Modula-2 (Exec)
Hex	Dez	struct Semaphore {	Semaphore = RECORD
0000	0	struct MsgPort sm_MsgPort;	msgPort:MsgPort;
0022	34	WORD sm_Bids;	bids:INTEGER;
0024	36	};	END;

Assembler — Syntax: Strukturname = *SM*.
Es ist nur *SM_BIDS* definiert.

Datenstrukturenreferenz — Keine.

Routinenreferenz — »Exec«: *Procure*, *Vacate*.

Beschreibung

Diese Datenstruktur dient zur Koordinierung des Zugriffs mehrerer Tasks auf ein beliebiges Objekt (ähnlich wie die *SignalSemaphore*-Datenstruktur). Vor dem Zugriff muß jedes Task durch einen *Procure*-Aufruf auf einen entsprechenden *Semaphore* prüfen, ob schon jemand anderes auf das Objekt zugreift. Ist dies der Fall, so gibt *Procure FALSE* zurück. Jede Anfrage wird in Form der an *Procure* übergebenen *Message* am Ende der Messageliste in *sm_MsgPort* gespeichert. Wenn der Task, der gerade an der Reihe ist, seinen Zugriff beendet, und dies durch *Vacate* signalisiert, wird die erste in der Liste stehende *Message* mit *Reply* beantwortet. Dadurch wird dem Urheber dieser Message die Zugriffsberechtigung mitgeteilt. Der Urheber wird durch den in der *Message*-Datenstruktur eingetragenen *MsgPort* (in *mn_ReplyPort*) identifiziert.

❑ **sm_MsgPort 0x0000 (0)**

Eine eingebundene *MsgPort*-Datenstruktur, die zur sequentiellen Speicherung der *Message*-Datenstrukturen Zugriffsanfragen dient.

❑ **sm_Bids 0x0022 (34)**

Jedesmal, wenn der Benutzer, der gerade das Zugriffsrecht hat *Procure* aufruft, wird dieser Zähler inkrementiert. Wird dagegen *Vacate* aufgerufen, so wird er dekrementiert. Das Objekt wird erst dann für andere freigegeben, wenn *sm_Bids* 0 ist.

SemaphoreRequest

Offset		C (exec/semaphores.h)	Modula-2 (Exec)
Hex	Dez	struct SemaphoreRequest {	SemaphoreRequest = RECORD
0000	0	struct MinNode sr_Link;	link:MinNode;
0008	8	struct Task *sr_Waiter;	waiter:TaskPtr;
000C	12	};	END;

Assembler Syntax: Strukturname = *SSR*
Es ist nur *SSR_WAITER* definiert.

Datenstrukturenreferenz »Exec«: *SignalSemaphore.*

Routinenreferenz Keine.

3

Beschreibung

Die *SemaphoreRequest*-Datenstruktur wird von Exec benutzt, um eine Zugriffsanforderung an einen *SignalSemaphore* zu speichern. Alle Anfragen, die an einen *SignalSemaphore* gerichtet werden und noch nicht erledigt wurden, werden in einer verketteten Liste solcher Datenstrukturen gespeichert. Sie werden über das *sr_Link*-Feld zu einer *MinList*-Liste verbunden.

- **sr_Link 0x0000 (0)** Dient zur Verkettung der einzelnen Datenstrukturen.
- **sr_Waiter 0x0008 (8)** Ein Adreßzeiger auf die *Task*-Datenstruktur des Task, der die Anforderung geschickt hat.

SignalSemaphore

Offset		C (exec/semaphores.h)	Modula-2 (Exec)
Hex	**Dez**	struct SignalSemaphore	SignalSemaphore =
		{	RECORD
0000	0	struct Node ss_Link;	link:Node;
000E	14	SHORT ss_NestCount;	nestCount:INTEGER;
0010	16	struct MinList ss_WaitQueue;	waitQueue:MinList;
001C	28	struct SemaphoreRequest	multipleLink:
		ss_MultipleLink;	SemaphoreRequest;
0028	40	struct Task *ss_Owner;	owner:TaskPtr;
002C	44	SHORT ss_QueueCount;	queueCount:INTEGER;
002E	46	};	END;

Assembler Syntax: Strukturname = *SS*
Definition der Parameter in Versalien, wobei *ss_Link* nicht definiert ist.

Datenstrukturenreferenz »Graphics«: *Layer, LayerInfo.*
»Intuition«: *IntuitionBase.*

Routinenreferenz »Exec«: *AddSemaphore, AttemptSemaphore, FindSemaphore, InitSemaphore, ObtainSemaphore, ReleaseSemaphore, RemSemaphore.*

Beschreibung

Diese Datenstruktur dient zur Koordinierung des Zugriffs mehrerer Tasks auf ein beliebiges Objekt (ähnlich wie die *Semaphore*-Datenstruktur). Ein solcher *SignalSemaphore* kann mit der Routine *InitSe-*

maphore eingerichtet werden. Ein Task, der auf das von der *SignalSemaphore*-Datenstruktur beschützte Objekt zugreifen will, siganlisiert dies mit *ObtainSemaphore* bzw. *AttemptSemaphore*. Nach beendetem Zugriff wird dann *ReleaseSemaphore* aufgerufen, damit das Objekt für andere Benutzer zugänglich wird. Dieser Mechanismus wird von Intuition genutzt, um die Layers (Bestandteile der Videoanzeige) zu verwalten.

- **ss_Link 0x0000 (0)** — Mit Hilfe dieser eingebundenen *Node*-Datenstruktur können mehrere *SignalSemaphore*-Datenstrukturen zu einer Liste verkettet werden.
- **ss_NestCount 0x000E (14)** — Falls ein Benutzer, der bereits das Zugriffsrecht hat, weitere Anforderungen schickt, wird dieser Zähler inkrementiert.
- **ss_WaitQueue 0x0010 (16)** — In dieser Liste werden die *SemaphoreRequest*-Datenstrukturen abgelegt, in denen sich die vorliegenden Anfragen befinden.
- **ss_MultipleLink 0x001C (28)** — Wörtlich übersetzt: »Mehrfache Verbindung« ???
- **ss_Owner 0x0028 (40)** — Ein Adreßzeiger auf die *Task*-Datenstruktur, des Tasks, der diesen *SignalSemaphore* eingerichtet hat.
- **ss_QueueCount 0x002C (44)** — Die Anzahl der in *WaitQueue* vorliegenden Anfragen.

SoftIntList

Offset		C (exec/interrupts.h)	Modula-2 (Exec)
Hex	**Dez**	struct SoftIntList	SoftIntList =
		{	RECORD
0000	0	struct List sh_List;	list:List;
000E	14	UWORD sh_Pad;	pad:WORD;
0010	16	};	END;

Assembler — Syntax: Strukturname = *SH*
Der Parameter *sh_List* ist nicht extra definiert.

Datenstrukturenreferenz — »Exec«: *ExecBase*.

Routinenreferenz — Keine.

Beschreibung

Eine Exec-interne Datenstruktur, die als Listenheader einer Liste der Software-Interrupts benutzt wird. Eine solche Datenstruktur befindet sich in der *ExecBase*.

- **sh_List 0x0000 (0)** — Eine eingebundene *List*-Datenstruktur, die als Listenheader benutzt wird.
- **sh_Pad 0x000E (14)** — Ein Füllbytechen.

3

Task

Offset		C (exec/tasks.h)	Modula-2 (Exec)
Hex	Dez	extern struct Task	Task =
		{	RECORD
0000	0	struct Node tc_Node;	node:Node;
000E	14	UBYTE tc_Flags;	flags:TaskFlagSet;
000F	15	UBYTE tc_State;	state:TaskState;
0010	16	BYTE tc_IDNestCnt;	idNestCnt:Byte;
0011	17	BYTE tc_TDNestCnt;	tdNestCnt:Byte;
0012	18	ULONG tc_SigAlloc;	sigAlloc:LONGSET;
0016	22	ULONG tc_SigWait;	sigWait:LONGSET;
001A	26	ULONG tc_SigRecvd;	sigRecvd:LONGSET;
001E	30	ULONG tc_SigExcept;	sigExcept:LONGSET;
0022	34	UWORD tc_TrapAlloc;	trapAlloc:BITSET;
0024	36	UWORD tc_TrapAble;	trapAble:BITSET;
0026	38	APTR tc_ExceptData;	exceptData:ADDRESS;
002A	42	APTR tc_ExceptCode;	exceptCode:PROC;
002E	46	APTR tc_TrapData;	trapData:ADDRESS;
0032	50	APTR tc_TrapCode;	trapCode:PROC;
0036	54	APTR tc_SPReg;	spReg:ADDRESS;
003A	58	APTR tc_SPLower;	spLower:ADDRESS;
003E	62	APTR tc_SPUpper;	spUpper:ADDRESS;
0042	66	VOID (*tc_Switch)();	switch:PROC;
0046	70	VOID (*tc_Launch)();	launch:PROC;
004A	74	struct List tc_MemEntry;	memEntry:List;
0058	88	APTR tc_UserData;	userData:ADDRESS;
005C	92	};	END;

Assembler Syntax: Strukturname = *TC*

Definition der einzelnen Parameter in Versalien (bis auf *TC_UserData*), wobei *tc_Node* nicht extra definiert ist.

Datenstrukturenreferenz
»Dos«: *Process.*
»Exec«: *ExecBase, MsgPort, SignalSemaphore.*
»Graphics«: *GfxBase.*
»PrtBase«: *PrinterData.*

Routinenreferenz
»Exec«: *AddTask, FindTask, RemTask, SetTaskPri, Signal.*
»ExecSupoort«: *CreateTask, DeleteTask.*

Beschreibung

Diese Datenstruktur beschreibt einen Task. Sie enthält alle Informationen, die das Multitasking für das Ein- und Ausschalten sowie die Behandlung der Traps und Exceptions braucht.

- **tc_Node 0x0000 (0)** Eine gewöhnliche *Node*-Datenstruktur, die zur Verkettung mehrerer Tasks zu einer Liste (z.B. in der *ExecBase*-Datenstruktur) dient.

- **tc_Flags 0x000E (14)** Dies sind Flags, die den momentanen Zustand des Task beschreiben. Es ist jede Kombination der folgenden Bits zugelassen:

C	Modula-2	Hex	Dez	Beschreibung
TF_PROCTIME	procTime	01	1	???
TF_STACKCHK	stackChk	10	16	Der Taskstack wird überprüft.
TF_EXCEPT	exception	20	32	Es ist eine Exception aufgetreten.
TF_SWITCH	switch	40	64	Der Task verliert den Prozessor.
TF_LAUNCH	launch	80	128	Der Task wird gerade eingeschaltet.

tc_State 0x000F (15) Diese Flags beschreiben den Task näher. Es sind Kombinationen folgender Werte zugelassen:

C	Modula-2	Dez	Beschreibung
TS_INVALID	invalid	0	Der Task ist ungültig.
TS_ADDED	added	1	Der Task wurde dem System gerade hinzugefügt.
TS_RUN	run	2	Der Task läuft gerade.
TS_READY	ready	3	Der Task ist bereit, wartet auf den Prozessor.
TS_WAIT	wait	4	Der Task wartet auf ein Signal. Er braucht also bei der Vergabe der Prozessorzeit nicht berücksichtigt zu werden.
TS_EXCEPT	except	5	Der Task behandelt gerade eine Exception.
TS_REMOVED	removed	6	Der Task wurde vom System entfernt.

- **tc_IDNestCnt 0x0010 (16)** Ein Zähler für die Anzahl der *Disable*-(wird inkrementiert) und *Enable*-(wird dekrementiert)Aufrufe. Erst wenn *ID_NestCnt* 0 ist, wird ein Interrupt zugelassen.

- **tc_TDNestCnt 0x0011 (17)** Ein Zähler für die Anzahl der *Forbid*-(wird inkrementiert) und *Permit*-(wird dekrementiert)Aufrufe. Erst wenn *TD_NestCnt* 0 ist, wird das Multitasking wieder eingeschaltet, der Task kann also den Prozessor verlieren.

- **tc_SigAlloc 0x0012 (18)** Jedes gesetzte Bit dieses Feldes bedeutet, daß das Signal mit der gleichen Nummer allokiert wurde.
- **tc_SigWait 0x0016 (22)** Jedes gesetzte Bit in diesem Feld entspricht einem Signal, auf dessen Ankunft der Task wartet. Sobald nur eines dieser Signale emfangen wird, wird der Wait-Zustand beendet.
- **tc_SigRecvd 0x001A (26)** Die Nummern der hier gesetzten Bits entsprechen den Nummern der Signale, die empfangen wurden.
- **tc_SigExcept 0x001E (30)** Die Nummern der hier gesetzten Bits entsprechen den Nummern der Signale, die eine Exception auslösen.
- **tc_TrapAlloc 0x0022 (34)** Die Nummern der hier gesetzten Bits entsprechen den Nummern der Traps, die allokiert wurden. Dies heißt, daß beim Auftreten des entsprechenden Prozessortraps die in Trapcode angegebene Funktion aufgerufen wird.
- **tc_TrapAble 0x0024 (36)** Die Nummern der hier gesetzten Bits entsprechen den Nummern aktiver Traps.
- **tc_ExceptData 0x0026 (38)** Ein Adreßzeiger auf den Datenbereich der Exceptionroutine. Diese Adresse wird an den Exceptionhandler in A1 übergeben.
- **tc_ExceptCode 0x002A (42)** Ein Adreßzeiger auf die Routine, die beim Auftreten einer Exception aufgerufen wird. Sie bekommt in D0 die Nummer des für die Exception verantwortlichen Signals. Dieser Wert muß vor dem Verlassen des Handlers mit RTS wieder in D0 geladen werden.
- **tc_TrapData 0x002E (46)** Ein Adreßzeiger auf den Datenbereich der Trap-Routine.
- **tc_TrapCode 0x0032 (50)** Ein Adreßzeiger auf den Code des Trap-Handlers. Er muß in dem Supervisormodus laufen können.
- **tc_SPReg 0x0036 (54)** Dies ist der Stackpointer des Task.
- **tc_SPLower 0x003A (58)** Dies ist die Adresse der unteren Grenze des Task-eigenen Stacks.
- **tc_SPUpper 0x003E (62)** Dies ist die Adresse der oberen Grenze des Task-eigenen Stacks.
- **tc_Switch 0x0042 (66)** Ein Adreßzeiger auf die Routine, die aufgerufen wird, wenn ein Task ausgeschaltet wird.
- **tc_Launch 0x0046 (70)** Ein Adreßzeiger auf eine Routine, die beim Einschalten des Tasks aufgerufen wird.
- **tc_MemEntry 0x004A (74)** Die Liste der für den Task allokierten Speicherbereiche. Beim Entfernen des Task werden alle diese Bereiche freigegeben.
- **tc_UserData 0x0058 (88)** Dieser Adreßzeiger steht dem Programmierer für eigene Zwecke zur Verfügung.

Unit

Offset		C (exec/devices.h)	Modula-2 (Exec)
Hex	Dez	struct Unit	Unit =
		{	RECORD
0000	0	struct MsgPort *unit_MsgPort;	msgPort:MsgPortPtr;
0004	4	UBYTE unit_flags;	flags:UnitFlagSet;
0005	5	UBYTE unit_pad;	pad:BYTE;
0006	6	UWORD unit_OpenCnt;	openCnt:CARDINAL;
0008	8	};	END;

Assembler Syntax: Strukturname = *UNIT*.

Definition der einzelnen Parameter in Versalien, wobei *unit_MsgPort* nicht definiert ist.

Datenstrukturenreferenz
»Clipboard«: *IOClipboard.*
»Exec«: *IORequest, IOStdReq.*
»Trackdisk«: *TDU_PublicUnit.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur wird bei der Verwaltung der Devices und Resources benutzt, um ein bestimmtes Gerät zu identifizieren. Dies kann ein echtes Hardwaregerät wie z.B. ein konkretes Laufwerk, oder einfach eine abstrakte Einheit sein (z.B. bei dem Clipboard-Device).

- **unit_MsgPort 0x0000 (0)** Ein *MsgPort*, an dem Anfragen (Messages) an dieses Gerät gerichtet und in der Messageliste bis zur Bearbeitung gespeichert werden.
- **unit_Flags 0x0004 (4)** Dies sind geräteinterne Flags. Zwei Flags sind für dieses Feld vordefiniert:

C	Modula-2	Dez	Beschreibung
UNITTF_ACTIVE	active	1	Das Gerät ist aktiv.
UNITTF_INTASK	inTask	2	Das Gerät ist gerade beschäftigt ?

- **unit_pad 0x0005 (5)** Noch ein letztes Füllbyte.
- **unit_OpenCnt 0x0006 (6)** Dieses Feld gibt an, wie oft dieses Unit schon göffnet wurde (z.B. wie oft schon eine Trackdisk-Device auf ein konkretes Laufwerk geöffnet wurde).

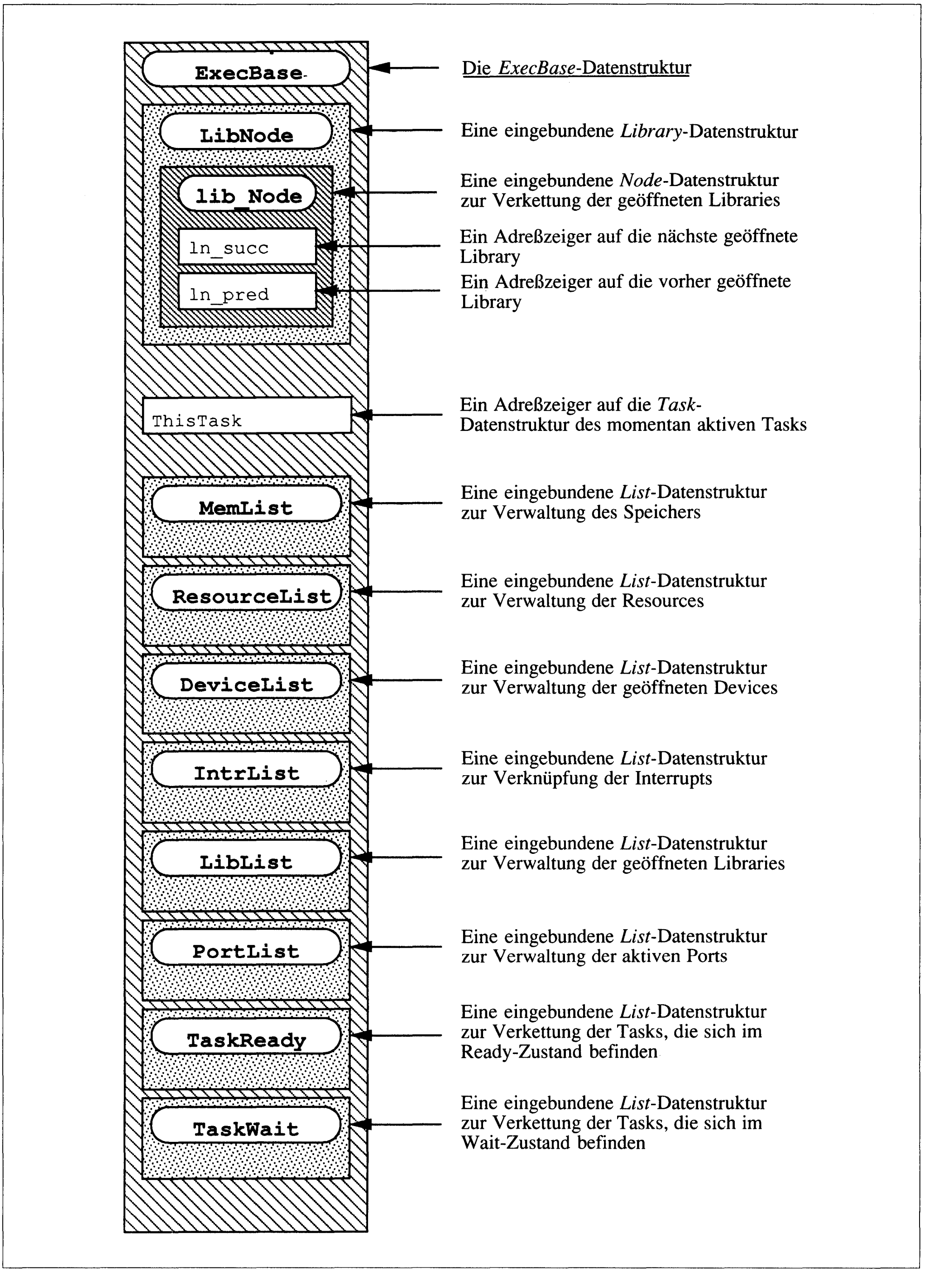


Abb. 3.1: Die ExecBase–Datenstruktur

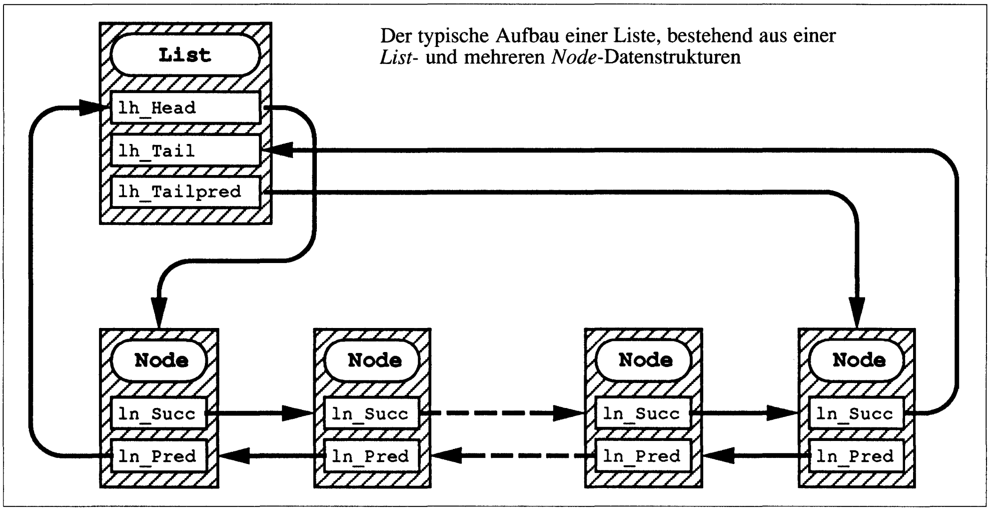


Abb. 3.2: Der Aufbau einer Liste

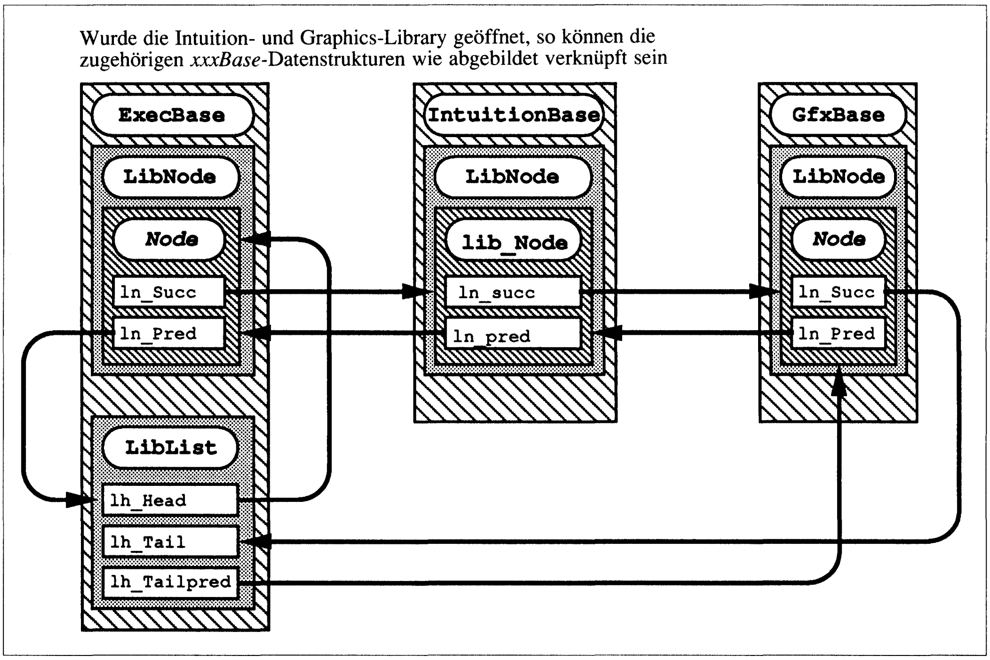


Abb. 3.3: Die Verknüpfung der Libraries

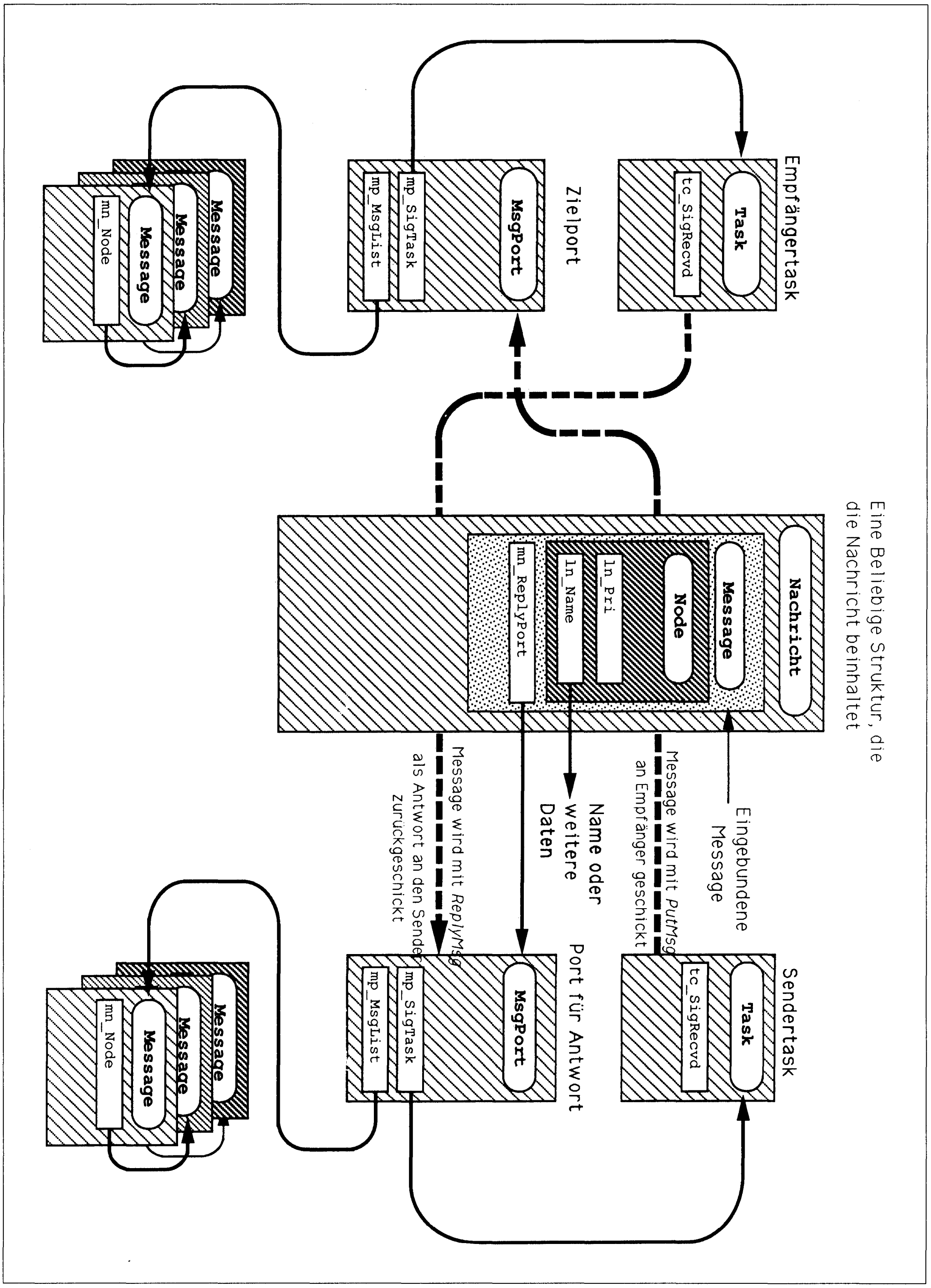


Abb. 3.4: Die Nachrichtenübermittlung zwischen Tasks

4 DIE INTUITION-LIBRARY

In dieser Library sind alle die Systemroutinen und Datenstrukturen zusammengefaßt, die mit der Benutzeroberfläche zu tun haben. In ihr sind nicht nur die Screens (von manchen auch Bildschirme genannt), Windows (Fenster kann man auch mal dazu sagen), Requester und Gadgets definiert (Commodores »Sales Support Information« betreffend der Übersetzungsrichtlinien von Amiga-Begriffen sagt dazu übrigens Kommunikationsfenster und Symbolfeld), sondern auch einige einfache Routinen zur Ausgabe von Texten und Grafiken.

4.1 Wichtige Flags der Intuition-Library

In diesem Abschnitt werden alle die Flags der Intuition-Library genauer beschrieben, die in mehreren Datenstrukturen verwendet werden.

Die IDCMP-Flags

(intuition/intuition.h) (Intuition)

C	Modula-2	Hex
SIZEVERIFY	sizeVerify	00000001

Eine Meldung über das Verändern der Größe des Fensters, jedoch wartet »Intuition« mit der Ausführung, bis auf die Message durch die Routine *ReplyMsg* geantwortet wurde.

C	Modula-2	Hex
NEWSIZE	newSize	00000002

Eine Meldung darüber, daß die Größe des Fensters bereits verändert wurde.

C	Modula-2	Hex
REFRESHWINDOW	refreshWindow	00000004

Eine Meldung über die Notwendigkeit, einen Refresh durchzuführen.

C	Modula-2	Hex
MOUSEBUTTONS	mouseButtons	00000008

Eine Meldung über das Betätigen der Maustasten.

C	Modula-2	Hex
MOUSEMOVE	mouseMove	00000010

Eine Meldung über das Bewegen der Maus.

C	Modula-2	Hex
GADGETDOWN	gadgetDown	00000020

Eine Meldung über ein momentan angeklicktes Gadget.

GADGETUP	gadgetUp	00000040

4

Eine Meldung über ein wieder losgelassenes Gadget.

REQSET	reqSet	00000080

Eine Meldung über die Öffnung eines Requesters.

MENUPICK	menuPick	00000100

Eine Meldung über das Drücken der rechten Maustaste.

CLOSEWINDOW	closeWindow	00000200

Eine Meldung über das Anklicken des Close-Gadgets.

RAWKEY	rawKey	00000400

Eine Tastaturmeldung im Raw-Code. In der *IntuiMessage*-Datenstruktur steht bei einer Systemnachricht der Raw-Code der gedrückten Taste in dem Feld Code.

REQVERIFY	reqVerify	00000800

Eine Meldung über das Öffnen eines Requesters, jedoch wartet »Intuition« mit der Ausführung, bis auf die Message durch die Routine *ReplyMsg* geantwortet wurde.

REQCLEAR	reqClear	00001000

Eine Meldung über das Schließen des letzten Requesters.

MENUVERIFY	menuVerify	00002000

Eine Meldung über das Zeichnen eines Pull-down-Menüs, jedoch wartet »Intuition« mit der Darstellung des Menüs, bis auf die Message durch die Routine *ReplyMsg* geantwortet wurde.

C	Modula-2	Hex
NEWPREFS	newPrefs	00004000

Eine Meldung über den Versuch des Benutzers, die Preferences-Einstellungen zu verändern.

DISKINSERTED	diskInserted	00008000

Eine Meldung über das Einlegen einer Diskette in ein Laufwerk.

DISKREMOVED	diskRemoved	00010000

Eine Meldung über das Entfernen einer Diskette aus einem Laufwerk.

WBENCHMESSAGE	wbenchMessage	00020000

Eine Meldung der Workbench.

ACTIVEWINDOW	activeWindow	00040000

Eine Meldung über das Aktivieren dieses Fensters.

INACTIVEWINDOW	inactiveWindow	00080000

Eine Meldung über das Inaktivieren dieses Fensters.

DELTAMOVE	deltaMove	00100000

Die Mausbewegungen werden relativ zu der letzten Mausposition angegeben. Dies geschieht auch, wenn sich der Zeiger nicht mehr über dem Fenster befindet.

VANILLAKEY	vanillaKey	00200000

Eine Tastaturmeldung im ASCII-Code. In der *IntuiMessage*-Datenstruktur steht bei einer Systemnachricht der ASCII-Code der gedrückten Taste gemäß der momentanen Keymap in dem Feld *Code*.

INTUITICKS	intuiTicks	00400000

Eine ständige aktuelle Systemnachricht. »Intuition« hält dabei bis zu zehnmal pro Sekunde eine *IntuiMessage*-Datenstruktur bereit, die jedoch nicht gepuffert wird. Eine neue Nachricht gibt es nur, wenn auf die vorherige durch die Routine *ReplyMsg* geantwortet wurde.

4

C	Modula-2	Hex
LONELYMESSAGE	lonelyMessage	80000000

Ein einsames Systeminternes Bit.

Datenstrukturenreferenz »Intuition«: *IntuiMessage*, *NewWindow*.

Beschreibung

4

Die *IDCMP-Flags* gehören zu dem »Intuition Direct Comunication Message Port«, einem besonderen Messageport, über den Informationen über die Windows gesendet werden. In der *NewWindow*-Datenstruktur definieren diese Flags, bei welchen Ereignissen eine Message losgeschickt wird. In der *IntuiMessage*-Datenstruktur dagegen definieren diese Flags, welches Ereignis eingetreten ist. Wird zum Beispiel beim Öffnen eines Windows *SIZEVERIFY* gesetzt, so wird jedesmal, wenn der Benutzer die Größe dieses Fensters verändert, eine *IntuiMessage*-Datenstruktur bereitgestellt, bei der in *Class SIZEVERIFY* gesetzt ist.

Die Screen-Flags

(intuition/intuition.h) (Intuition)

Definierte Konstanten:

C	Modula-2	Hex
SCREENTYPE		000F

Das untere Nibbel enthält die Informationen über die Art des Screens.

CUSTOMSCREEN	customScreen	000F

Der zugehörige Bildschirm ist kein Workbench-Screen.

STDSCREENHEIGHT	stdScreenHeight	FFFF

Der zugehörige Bildschirm hat die Standardhöhe von 256 Zeilen.

Definierte Bits

C	Modula-2	Hex
WBENCHSCREEN	wbenchScreen	0001

Der zugehörige Bildschirm ist ein Workbench-Screen.

SHOWTITLE	showTitle	0010

Die Titelleiste des zugehörigen Bildschirms wird mitausgegeben.

BEEPING	beeping	0020

Der zugehörige Bildschirm kann blinken bzw. blinkt gerade.

CUSTOMBITMAP	customitMap	0040

Der zugehörige Bildschirm hat eine eigene Bitmap.

SCREENBEHIND	screenBehind	0080

Beim Öffnen ist der Bildschirm im Hintergrund.

SCREENQUIET	screenQuiet	0100

Zu diesem Bildschirm gehören keine Gadgets.

Datenstrukturenreferenz »Intuition«: *NewScreen, NewWindow, Screen.*

Beschreibung

Die Screen-Flags enthalten Informationen über die Eigenschaften eines Bildschirms.

Die Window-Flags

(intuition/intuition.h) (Intuition)

Definierte Konstanten

C	Modula-2	Hex
SMART_REFRESH		00000000

Ein Refresh wird nur notwendig, wenn das Fenster vom Benutzer verkleinert und anschließend wieder vergrößert wurde.

4

REFRESHBITS		000000C0

Die Bits, die den Refresh-Modus für dieses Fenster festlegen.

OTHER_REFRESH	otherRefresh	000000C0

Refresh für *SUPER_BITMAP*-Windows, also Fenster mit eigenen Bitmaps.

SUPER_UNUSED	superUnused	FCFC0000

Eine Kombination aller unbenutzten Bits der Window-Flags ergibt die »Super-Unbenutzten-Flags«.

Definierte Bits

C	Modula-2	Hex
WINDOWSIZING	windowSizing	00000001

Das Sizing-Gadget erscheint in der unteren rechten Ecke (die Fenstergröße kann verändert werden).

WINDOWDRAG	windowDrag	00000002

Die Titelleiste wird zu einem Drag-Gadget (das Fenster ist verschiebbar).

WINDOWDEPTH	windowDepth	00000004

Die Depth-Gadgets erscheinen in der rechten, oberen Ecke (das Fenster kann in den Vorder- oder Hintergrund geschaltet werden).

WINDOWCLOSE	windowClose	00000008

Das Close-Gadget erscheint in der linken oberen Ecke. Wird es durch den Mauspfeil aktiviert, schließt sich das Fenster jedoch nicht automatisch. Es wird lediglich eine entsprechende Systemnachricht bereitgestellt.

C	Modula-2	Hex
SIZEBRIGHT	sizeBRight	00000010

Das Sizing-Gadget gehört zum rechten Fensterrand (voreingestellt).

SIZEBBOTTOM	sizeBBottom	00000020

Das Sizing-Gadget gehört zum unteren Fensterrand.

4

SIMPLE_REFRESH	simpleRefresh	00000040

Es wird kein zusätzlicher Speicher für den Refresh bereitgehalten. Dadurch muß der Inhalt des Fensters nach Überlappungen mit anderen Fenstern oder nach einer Vergrößerung jedesmal neu aufgebaut werden.

SUPER_BITMAP	superBitMap	00000080

Es existiert eine eigene Bitmap, die groß genug ist, den gesamten Fenster-Inhalt aufzunehmen. Bei Überlappungen mit anderen Fenstern oder bei Veränderungen der Größe ist somit kein Refresh notwendig.

BACKDROP	backDrop	00000100

Das Fenster bleibt immer im Hintergrund.

REPORTMOUSE	reportMouse	00000200

Es werden Systemnachrichten über die Mausbewegungen innerhalb dieses Fensters bereitgestellt.

GIMMEZEROZERO	gimmeZeroZero	00000400

Dieses Fenster ist ein GimmeZeroZero-Window (Koordinatenursprung liegt in der inneren linken, oberen Ecke, von den Höhen- und Breitenwerten des Fensters werden also die Rahmen abgezogen).

BORDERLESS	borderless	00000800

Das Fenster erhält keinen Rahmen. ACHTUNG: Sind System-Gadgets aktiviert, so werden zugehörige Ränder trotzdem ausgegeben.

C	Modula-2	Hex
ACTIVATE	activate	00001000

Das Fenster wird sofort nach dem Öffnen aktiviert. ACHTUNG: Macht der Benutzer gerade eine Eingabe, so wird diese auf das neue Fenster umgeleitet.

WINDOWACTIVE	windowActive	00002000

Das Fenster ist momentan aktiv.

INREQUEST	inRequest	00004000

Das Fenster ist durch mindestens einen Requester gesperrt.

MENUSTATE	menuState	00008000

Ein Menüpunkt wird gerade ausgewählt.

RMBTRAP	rmbTrap	00010000

Über das Drücken der rechten Maustaste wird eine Message erstellt, eine Menüauswahl ist daher nicht möglich.

NOCAREREFRESH	noCareRefresh	00020000

Es werden keine Systemnachrichten über die Notwendigkeit eines Refreshs bereitgestellt.

WINDOWREFRESH	windowRefresh	01000000

An dem Fenster wird momentan ein Refresh durchgeführt.

WBENCHWINDOW	wbenchWindow	02000000

Das Fenster ist auf der Workbench geöffnet.

WINDOWTICKED	windowTicked	04000000

Systeminternes Bit.

Datenstrukturenreferenz »Intuition«: *NewWindow*, *NewScreen*.

Beschreibung

Die Window-Flags enthalten Informationen über die Eigenschaften eines Fensters.

4.2 Die Datenstrukturen der Intuition-Library

BoolInfo

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	**Dez**	struct BoolInfo	BoolInfo =
		{	RECORD
0000	0	USHORT Flags;	flags:BITSET;
0002	2	UWORD *Mask;	mask:ADDRESS;
0006	6	ULONG Reserved;	reserved:LONGCARD;
000A	10	};	END;

Assembler Präfix: *bi_*

Datenstrukturenreferenz »Intuition«: *Gadget.*

Routinenreferenz Keine.

Beschreibung

Dies ist eine zusätzliche Datenstruktur für Boolean-Gadgets (für ja/nein Entscheidungen wie »OK« oder »Cancel«) und sogenannten Boolean-Toggle-Gadgets (Ein- und Ausschalten von bestimmten Funktionen).

❑ **Flags 0x0000 (0)** Es ist nur folgende Konstante vereinbart:

C	Modula-2	Hex	Beschreibung
BOOLMASK	boolMask	0001	Ist immer gesetzt, da bis jetzt dies die einzige Definition ist.

❑ **Mask 0x0002 (2)** Ein Adreßzeiger auf eine *Image*-Datenstruktur, die das Grafiksymbol beschreibt, das bei der Auswahl dieses Gadgets auf dem Display erscheint.

❑ **Reserved 0x0006 (6)** Reservierter Adreßzeiger für zukünftige Erweiterungen. In den vorliegenden Versionen immer auf *NULL* setzen.

4

Border

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	**Dez**	struct Border	Border =
		{	RECORD
0000	0	SHORT LeftEdge;	leftEdge:INTEGER;
0002	2	SHORT TopEdge;	topEdge:INTEGER;
0004	4	UBYTE FrontPen;	frontPen:UByte;
0005	5	UBYTE BackPen;	backPen:UByte;
0006	6	UBYTE DrawMode;	drawMode:DrawModeSet;
0007	7	BYTE Count;	count:UByte;
0008	8	SHORT *XY;	xy:ADDRESS;
000C	12	struct Border *NextBorder;	nextBorder:BorderPtr;
0010	16	};	END;

Assembler Präfix: *bd_*

Datenstrukturenreferenz »Intuition«: *Border, Requester.*

Routinenreferenz »Intuition«: *DrawBorder.*

Beschreibung

In dieser Datenstruktur werden die Informationen über einen zu zeichnenden Rahmen für die *DrawBorder*-Routine bereitgehalten. Dabei handelt es sich eigentlich um einen beliebigen Linienzug, mit dem man natürlich auch einen rechteckigen Rahmen zeichnen kann.

❑ **LeftEdge 0x0000 (0)** Die zu der x-Koordinate, die beim Aufruf in der *DrawBorder*-Routine angegeben wird, relative x-Koordinate des Basispunktes, der für das Zeichnen dieses Linienzuges als Ursprung dient.

❑ **TopEdge 0x0002 (2)** Die zu der y-Koordinate, die beim Aufruf in der *DrawBorder*-Routine angegeben wird, relative y-Koordinate des Basispunktes, der für das Zeichnen dieses Linienzuges als Ursprung dient.

❑ **FrontPen 0x0004 (4)** Die Nummer des Farbregisters, dessen Inhalt als Vordergrundfarbe verwendet wird.

❑ **BackPen 0x0005 (5)** Die Nummer des Farbregisters, dessen Inhalt als Hintergrundfarbe verwendet wird.

❑ **DrawMode 0x0006 (6)** Der Zeichenmodus, mit dem der Rahmen gezeichnet werden soll. Zur Verfügung stehen *JAM1* (Zeichnen mit der Vordergrundfarbe) und *XOR* (Zeichnen durch D invertieren).

❑ **Count 0x0007 (7)** Anzahl der Koordinatenpaare, die in *XY* gespeichert sind.

❑ **XY 0x0008 (8)**

Dies ist ein Adreßzeiger auf eine Liste mit x- und y-Koordinatenpaaren. Jedes Koordinatenpaar definiert einen Eckpunkt des Linienzuges, und zwar relativ zu dem durch *LeftEdge* und *TopEdge* definierten Basispunkt.

❑ **NextBorder 0x000C (12)**

Sollen mit dem gleichen Aufruf noch mehr Linienzüge gezeichnet werden, so können über diesen Adreßzeiger mehrere *Border*-Datenstrukturen miteinander verknüpft werden. Steht hier *NULL*, so ist das Ende der Liste erreicht.

FatIntuiMessage

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	**Dez**	struct FatIntuiMessage	FatIntuiMessage =
		{	RECORD
0000	0	struct IntuiMessage;	intuiMessage:IntuiMessage;
0034	52	ULONG PrevKeys;	prevKeys:LONGCARD;
0038	56	};	END;

Assembler

Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz

Keine.

Routinenreferenz

Keine.

> **Beschreibung**

Bei dieser systeminternen Datenstruktur handelt es sich um eine *IntuiMessage*-Datenstruktur, die um einen Parameter erweitert wurde.

❑ **IntuiMessage 0x0000 (0)**

Einbindung einer vollständigen *IntuiMessage*-Datenstruktur.

❑ **PrevKeys 0x0034 (52)**

Interner Systemparameter um den die *IntuiMessage*-Datenstruktur erweitert wurde – aber was für einer?

Gadget

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	**Dez**	struct Gadget	Gadget =
		{	RECORD
0000	0	struct Gadget *NextGadget;	nextGadget:GadgetPtr;
0004	4	SHORT LeftEdge;	leftEdge:INTEGER;

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
0006	6	SHORT TopEdge;	topEdge:INTEGER;
0008	8	SHORT Width;	width:INTEGER;
000A	10	SHORT Height;	height:INTEGER;
000C	12	USHORT Flags;	flags:GadgetFlagSet;
000E	14	USHORT Activation;	activation:ActivationFlagSet;
0010	16	USHORT GadgetType;	gadgetType:CARDINAL;
0012	18	APTR GadgetRender;	gadgetRender:ADDRESS;
0016	22	APTR SelectRender;	selectRender:ADDRESS;
001A	26	struct IntuiText *GadgetText;	gadgetText:IntuiTextPtr;
001E	30	LONG MutualExclude;	mutualExclude:LONGSET;
0022	34	APTR SpecialInfo;	specialInfo:ADDRESS;
0026	38	USHORT GadgetID;	gadgetID:INTEGER;
0028	40	APTR UserData;	userData:ADDRESS;
002C	44	};	END;

Assembler Präfix: *gg_*

Datenstrukturenreferenz »Intuition«: *DiskObjekt*, *Gadget*, *GadgetInfo*, *IntuitionBase*, *NewScreen*, *NewWindow*, *Requester*, *Screen*, *Window*.

RoutinenReferenz »Intuition«: *ActivateGadget*, *AddGadget*, *AddGList*, *ModifyProp*, *NewModifyProp*, *OffGadget*, *OnGadget*, *RefreshGadgets*, *RefreshGList*, *RemoveGadget*, *RemoveGList*.

Beschreibung

Die Gadgets sind einem Window zugeordnete Felder, die durch Anklicken mit der Maus eine Aktion auslösen können. Neben den bekannten *System*-Gadgets, wie *Close-*, *Depth-* und *Sizing*-Gadget, können zu jedem Displayelement beliebige Gadgets über diese Datenstruktur definiert werden. Dabei gibt es drei verschiedene Arten: *Boolean-*, *Proportional-* und *String*-Gadgets (siehe auch: *BoolInfo-*, *PropInfo-* und *StringInfo*-Datenstruktur.

❑ **NextGadget 0x0000 (0)** Gehören mehrere *Gadget*-Datenstrukturen zu diesem Displayelement, so werden sie über diesen Adreßzeiger miteinander zu einer linearen Liste verknüpft. Das letzte Glied dieser Kette muß an dieser Stelle den Wert *NULL* haben.

❑ **LeftEdge 0x0004 (4)** Der Abstand des Gadgets vom linken Rand des zugehörigen Displayelements. Die Bedeutung dieses Wertes hängt von dem Bit *GRELRIGHT* aus *Flags* ab.

- **TopEdge 0x0006 (6)** Der Abstand des Gadgets vom oberen Rand des zugehörigen Displayelements. Die Bedeutung dieses Wertes hängt von dem Bit *GRELBOTTOM* aus *Flags* ab.

- **Width 0x0008 (8)** Die Breite des Gadgets. Die Bedeutung dieses Wertes hängt von dem Bit *GRELWIDTH* aus *Flags* ab.

- **Height 0x000A (10)** Die Höhe des Gadgets. Die Bedeutung dieses Wertes hängt von dem Bit *GRELHEIGHT* aus *Flags* ab.

- **Flags 0x000C (12)** Gesetzte Flags haben folgende Bedeutung:

C	Modula-2	Hex	Beschreibung
GADGHIGHBITS	gadgHighbits	0003	Für diese zwei Bits sind folgende 4 Möglichkeiten des Highlightnings definiert:
GADGHCOMP	gadgetHComp	0000	00: Invertieren aller Bits in diesem Gadget.
GADGHBOX	gadgHBox	0001	01: Gadget mit einem Auswahlkasten umranden.
GADGHIMAGE	gadgHImage	0002	10: Ausgabe einer anderen Image- oder Border-Datenstruktur.
GADGHNONE	gadgHNone	0003	11: Wird dieses Gadget ausgewählt, so wird sein Erscheinungsbild nicht verändert.
GADGIMAGE	gadgImage	0004	1: Gadget wird als »Image« ausgegeben. 0: Gadget wird als »Border« ausgegeben. Beachte auch: GadgetRender und SelectRender.
GRELBOTTOM	gRelBottom	0008	TopEdge ist relativ zum unteren Rand dieses Displayelements und nicht zum oberen.
GRELRIGHT	gRelRight	0010	LeftEdge ist relativ zum rechten Rand dieses Displayelements und nicht zum linken.
GRELWIDTH	gRelWidth	0020	Width ist kein absoluter Wert, sondern wird von Width des Displayelements abgezogen.
GRELHEIGHT	gRelHeight	0040	Height ist kein absoluter Wert, sondern wird von Height des Displayelements abgezogen.
SELECTED	selected	0080	Handelt es sich um ein »Boolean-Gadget«, so ist es zur Zeit eingeschaltet.

C	Modula-2	Hex	Beschreibung
GADGDISABLED	gadgDisabled	0100	Dieses Gadget ist inaktiviert und kann vom Benutzer nicht wieder aktiviert werden.

ACHTUNG: In Modula-2 sind *gadgHighbits*, *gadgHNone* und *gadgHComp* als Konstanten definiert.

❑ **Activation 0x000E (14)** Gesetzte Flags haben folgende Bedeutung:

4

C	Modula-2	Hex	Beschreibung
RELVERIFY	relVerify	0001	Gadget wird nur aktiv, wenn sich der Mauszeiger beim Drücken und Loslassen der linken Maustaste über dem Gadget befindet.
GADGIMMEDIATE	gadgImmediate	0002	Gadget wird bereits beim ersten »Anklicken« aktiv.
ENDGADGET	endGadget	0004	Ist das Displayelement ein Requester, so kann es nur geschlossen werden, wenn dieses Bit gesetzt ist.
FOLLOWMOUSE	followMouse	0008	Auch nach dem »Anklicken« eines Gadgets wird das Programm laufend über die Position des Mauszeigers informiert.
RIGHTBORDER	rightBorder	0010	Ist das Displayelement ein Window, so kann das Gadget durch Setzen eines
LEFTBORDER	leftBorder	0020	dieser Bits in den rechten, linken, oberen oder unteren Rand eingebun-
TOPBORDER	topBorder	0040	den werden, wobei die Breite des Fensterrands der Größe des Gadgets
BOTTOMBORDER	bottomBorder	0080	angeglichen wird.
TOGGLESELECT	toggleSelect	0100	Es handelt sich um ein Boolean-Toggle-Gadget, das ein- und ausgeschaltet werden kann.
STRINGCENTER	stringCenter	0200	Ist dies ein String-Gadget, so wird der Text zentriert ein- und ausgegeben.
STRINGRIGHT	stringRight	0400	Ist dies ein String-Gadget, so wird der Text rechtsbündig ein- und ausgegeben.
LONGINT	longint	0800	Die Eingabe wird als Zahl vom Typ Long interpretiert.

C	Modula-2	Hex	Beschreibung
ALTKEYMAP	altKeyMap	1000	Zur Eingabe eines Textes wird eine eigene Keymap verwendet.
BOOLEXTEND	boolExtend	2000	Zu diesem Boolean-Gadget gehört eine *BoolInfo*-Datenstruktur.

❑ **GadgetType 0x0010 (16)**

Folgende Konstanten definieren in dieser 2-Byte-Marke, um welche Art von Gadget es sich handelt:

C	Modula-2	Hex	Beschreibung
			Definition der Anwender-Gadgets:
BOOLGADGET	boolGadget	0001	Ein Boolean-Gadget.
GADGET0002	gadget0002	0002	Lückenfüllendes Bit.
PROPGADGET	propGadget	0003	Ein Proportional-Gadget.
STRGADGET	strGadget	0004	Ein String-Gadget.
			Definition der System-Gadgets:
SIZING	sizing	0010	Ein Sizing-Gadget; zum Verändern der Größe eines Fensters.
WDRAGGING	wDragging	0020	Ein Drag-Gadget; zum Verschieben von Fenstern.
SDRAGGING	sDragging	0030	Ein Drag-Gadget; zum Verschieben von Bildschirmen.
WUPFRONT	wUpFront	0040	Ein Depth-Gadget; um ein Fenster in den Vordergrund zu holen.
SUPFRONT	sUpFront	0050	Ein Depth-Gadget; um einen Bildschirm in den Vordergrund zu holen.
WDOWNBACK	wDownBack	0060	Ein Depth-Gadget; um ein Fenster in den Hintergrund zu bringen.
SDOWNBACK	sDownBack	0070	Ein Depth-Gadget; um einen Bildschirm in den Hintergrund zu bringen.
CLOSE	close	0080	Ein Close-Gadget; zum Schließen von Fenstern. Definition der Zugehörigkeit des Gadgets:
REQGADGET	reqGadget	1000	Das Gadget gehört zu einem Requester.
GZZGADGET	gzzGadget	2000	Das Gadget ist in einem GZZ-Border integriert.
SCRGADGET	scrGadget	4000	Das Gadget gehört zu einem Bildschirm.
SYSGADGET	sysGadget	8000	Das Gadget ist ein System-Gadget.
GADGETTYPE	gadgetType	FC00	Kombination aller Flaggs, die in GadgetType gesetzt werden können.

❑ **GadgetRender 0x0012 (18)**

Ein Adreßzeiger auf eine *Image*- oder *Border*-Datenstruktur, die zur grafischen Darstellung dieses Gadgets benutzt wird. Um welche Datenstruktur es sich hierbei letztlich handelt, wird durch das *GADGIMAGE*-Bit in *Flags* angegeben. Steht hier *NULL*, so erfolgt keine sichtbare Markierung des Gadgets.

❑ **SelectRender 0x0016 (22)**

Ein Adreßzeiger auf eine *Image*- oder *Border*-Datenstruktur, die zur grafischen Darstellung bei Auswahl dieses Gadgets benutzt wird. Diese Datenstruktur muß vom gleichen Typ sein, wie die durch *GadgetRender* definierte. Um bei der Auswahl des Gadgets aktiv zu werden, muß das *GADGHIMAGE*-Bit in *Flags* gesetzt sein.

❑ **GadgetText 0x001A (26)**

Soll ein Text in diesem Gadget ausgegeben werden, so muß hier ein Adreßzeiger auf eine *IntuiText*-Datenstruktur stehen. Die relativen Koordinaten der linken oberen Ecke des Textes beziehen sich dann auf die *LeftEdge*- und *TopEdge*-Werte dieses Gadgets. Steht hier *NULL*, so wird kein Text ausgegeben.

❑ **MutualExclude 0x001E (30)**

Dieser Parameter wird in den vorliegenden Versionen von »Intuition« noch nicht unterstüzt.

❑ **SpecialInfo 0x0022 (34)**

Ein Adreßzeiger auf die zugehörige *BoolInfo*-, *PropInfo*- oder *StringInfo*-Datenstruktur. Von welcher Art das Gadget nun wirklich ist, wird durch die Flags in *GadgetType* festgelegt.

❑ **GadgetID 0x0026 (38)**

Fortlaufende Nummer dieses Gadgets in der linearen Liste aller zu diesem Window miteinander verknüpften Gadgets.

❑ **UserData 0x0028 (40)**

Möchten Sie einem Gadget eigene Zusatzdaten zuordnen, so können Sie hier einen Adreßzeiger auf diese Daten eintragen.

GadgetInfo

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	**Dez**	struct GadgetInfo	GadgetInfo =
		{	RECORD
0000	0	struct GListEnv *gi_Environ;	environ:GListEnvPtr;
0004	4	struct Gadget *gi_Gadget;	gadget:GadgetPtr;
0008	8	struct IBox gi_Box;	box:IBox;
0010	16	struct IBox gi_Container;	container:IBox;
0018	24	struct Layer *gi_Layer;	layer:LayerPtr;
001C	28	struct IBox gi_NewKnob;	newKnob:IBox;
0024	36	};	END;

Assembler — Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz — »Intuition«: *IntuitionBase*.
»Workbench«: *DiskObject*.

Routinenreferenz — Keine.

Beschreibung

In dieser Datenstruktur werden zu einem Gadget Informationen über seine Umgebung abgelegt.

- **gi_Environ 0x0000 (0)** — Ein Adreßzeiger auf eine *GListEnv*-Datenstruktur, die die Umgebung des Gadgets noch näher beschreibt.
- **gi_Gadget 0x0004 (4)** — Ein Adreßzeiger auf die *Gadget*-Datenstruktur, auf die sich diese Informationen beziehen.
- **gi_Box 0x0008 (8)** — Einbindung einer *IBox*-Datenstruktur, in der die Dimensionen des Gadgets gespeichert sind.
- **gi_Container 0x0010 (16)** — Einbindung einer *IBox*-Datenstruktur, in der die Dimensionen des Containers gespeichert sind.
- **gi_Layer 0x0018 (24)** — Ein Adreßzeiger auf die zu diesem Gadget gehörige *Layer*-Datenstruktur.
- **gi_NewKnob 0x001C (28)** — Einbindung einer *IBox*-Datenstruktur, in der bei einem *Proportional*-Gadget die Dimensionen des Knopfes gespeichert sind.

4

GListEnv

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	**Dez**	struct GListEnv	GListEnv =
		{	RECORD
0000	0	struct Screen *ge_Screen;	screen:ScreenPtr;
0004	4	struct Window *ge_Window;	window:WindowPtr;
0008	8	struct Requester *ge_Requester;	requester:RequesterPtr;
000C	12	struct RastPort *ge_RastPort;	rastPort:RastPortPtr;
0010	16	struct Layer *ge_Layer;	layer:LayerPtr;
0014	20	struct Layer *ge_GZZLayer;	gzzLayer:LayerPtr;
0018	24	struct PenPair ge_Pens;	pens:PenPair;
001A	26	struct IBox ge_Domain;	domain:IBox;
0022	34	struct IBox ge_GZZdims;	gzzDims:IBox;
002A	42	};	END;

Assembler Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz »Intuition«: *GadgetInfo, IntuitionBase.*

Routinenreferenz Keine.

Beschreibung

In dieser Datenstruktur werden zu einer Liste von mehreren Gadgets weitere Informationen über ihre Umgebung abgelegt.

4

- **ge_Screen 0x0000 (0)** Ein Adreßzeiger auf die *Screen*-Datenstruktur, zu dem die Gadgets gehören.
- **ge_Window 0x0004 (4)** Gehören die Gadgets zu einem Window, so steht hier ein Adreßzeiger auf die entsprechende *Window*-Datenstruktur.
- **ge_Requester 0x0008 (8)** Gehören die Gadgets zu einem Requester, so steht hier ein Adreßzeiger auf die entsprechende *Requester*-Datenstruktur.
- **ge_RastPort 0x000C (12)** Ein Adreßzeiger auf die *RastPort*-Datenstruktur, deren Parameter bei der Darstellung der Gadgets verwendet werden.
- **ge_Layer 0x0010 (16)** Ein Adreßzeiger auf die zu den Gadgets gehörende *Layer*-Datenstruktur.
- **ge_GZZLayer 0x0014 (20)** Gehören die Gadgets zu einem GimmeZeroZero-Window, so steht hier der zweite Adreßzeiger auf die entsprechende *Layer*-Datenstruktur.
- **ge_Pens 0x0018 (24)** Einbindung einer *PenPair*-Datenstruktur, in der die Nummern der Farbregister stehen, deren Farbwerte bei der Darstellung der Gadgets verwendet werden.
- **ge_Domain 0x001A (26)** Einbindung einer *IBox*-Datenstruktur, in der die Position und Dimension des Bereiches angegeben sind, in dem sich alle diese Gadgets befinden. Die Positionsangaben dieses Rechtecks sind relativ zum zugehörigen Displayelement.
- **ge_GZZdims 0x0022 (34)** Ist das zugehörige Displayelement ein GimmeZeroZero-Window, so stehen in dieser eingebundenen *IBox*-Datenstruktur die entsprechend veränderten Werte aus *ge_Domain.*

IBox

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	Dez	struct IBox	IBox =
		{	RECORD
0000	0	SHORT Left;	left:INTEGER;
0002	2	SHORT Top;	top:INTEGER;
0004	4	SHORT Width;	width:INTEGER;
0006	6	SHORT Height;	height:INTEGER;
0008	8	};	END;

Assembler Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz »Intuition«: *GadgetInfo*, *GListEnv*, *IntuitionBase*.

Routinenreferenz Keine.

Beschreibung

Diese systeminterne Datenstruktur wird benutzt, um die Dimensionen eines Rechtecks in andere Datenstrukturen einzubinden.

- **Left 0x0000 (0)** Der Abstand des Rechtecks vom linken Rand des zugehörigen Displayelements.
- **Top 0x0000 (0)** Der Abstand des Rechtecks vom oberen Rand des zugehörigen Displayelements.
- **Width 0x0004 (4)** Die Breite des Rechtecks.
- **Height 0x0004 (4)** Die Höhe des Rechtecks.

Image

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct Image	Image =
		{	RECORD
0000	0	SHORT LeftEdge;	leftEdge:INTEGER;
0002	2	SHORT TopEdge;	topEdge:INTEGER;
0004	4	SHORT Width;	width:INTEGER;
0006	6	SHORT Height;	height:INTEGER;
0008	8	SHORT Depth;	depth:INTEGER;
000A	10	USHORT *ImageData;	imageData:ADDRESS;
000E	14	UBYTE PlanePick;	planePick:UByte;

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
000F	15	UBYTE PlaneOnOff;	planeOnOff:UByte;
0010	16	struct Image *NextImage;	nextImage:ImagePtr;
0014	20	};	END;

Assembler Präfix: *ig_*

Datenstrukturenreferenz »Intuition«: *Image*, *IntuitionBase*, *NewWindow*, *Window*

Routinenreferenz »Intuition«: *DrawImage*.

Beschreibung

In dieser Datenstruktur werden die Daten gespeichert, die für die Ausgabe eines beliebigen Grafiksymbols mit der Routine *DrawImage* nötig sind.

- **LeftEdge 0x0000 (0)** Die zu der x-Koordinate, die beim Aufruf in der *DrawImage*-Routine angegeben wird, relative x-Koordinate der oberen linken Ecke des Grafiksymbols.
- **TopEdge 0x0000 (2)** Die zu der y-Koordinate, die beim Aufruf in der *DrawImage*-Routine angegeben wird, relative y-Koordinate der oberen linken Ecke des Grafiksymbols.
- **Width 0x0004 (4)** Die Breite des Grafiksymbols.
- **Height 0x0006 (6)** Die Höhe des Grafiksymbols.
- **Depth 0x0008 (8)** Die Tiefe des Grafiksymbols (also die Anzahl der Bitplanes), von der die Anzahl der Farben abhängt.
- **ImageData 0x000A (10)** Dies ist ein Adreßzeiger auf die Grafikdaten, die sich im **Chip-RAM** befinden müssen. Die einzelnen Bitplanes werden dabei hintereinander abgespeichert.
- **PlanePick 0x000E (14)** Die Bits dieser 1-Byte-Maske geben an, in welche Bitplanes des Ziel-Rastports die Grafikdaten geschrieben werden. Ist ein Bit gesetzt, so bedeutet dies, daß die korrespondierende Bitplane beschrieben wird.
- **PlaneOnOff 0x000F (15)** Die einzelnen Bits dieser 1-Byte-Maske sind nur von Interesse, wenn das korrespondierende Bit in *PlanePick* gelöscht ist. In diesem Fall wird die zugehörige Bitplane in dem Bereich des Grafiksymbols mit 0 oder 1 gefüllt (ist also z.B. das dritte Bit in *PlanePick* gelöscht und in *PlaneOnOff* gesetzt, so wird die dritte Bitplane des Ziel-Rastports in dem Bereich des Grafiksymbols mit 1 gefüllt).

❑ **NextImage 0x0010 (16)**

Über diesen Adreßzeiger können mehrere *Image*-Datenstrukturen miteinander verknüpft werden, um so durch einen einzigen Aufruf der *DrawImage*-Routine mehrere Grafiksymbole in dem entsprechenden RastPort zu zeichnen. Das letzte Element der Liste muß an dieser Stelle wie immer den Wert *NULL* haben.

IntuiMessage

Offset Hex	Offset Dez	C (intuition/intuition.h)	Modula-2 (Intuition)
		struct IntuiMessage	IntuiMessage =
		{	RECORD
0000	0	struct Message ExecMessage;	execMessage:Message;
0014	20	ULONG Class;	class:IDCMPFlagSet;
0018	24	USHORT Code;	code:CARDINAL;
001A	26	USHORT Qualifier;	qualifier:QualifierSet;
001C	28	APTR IAddress;	iAddress:ADDRESS;
0020	32	SHORT MouseX;	mouseX:INTEGER;
0022	34	SHORT MouseY;	mouseY:INTEGER;
0024	36	ULONG Seconds;	seconds:LONGCARD;
0028	40	ULONG Micros;	micros:LONGCARD;
002C	44	struct Window *IDCMPWindow;	idcmpWindow:WindowPtr;
0030	48	struct IntuiMessage	specialLink:IntuiMessagePtr;
		*SpecialLink	
0034	52	};	END;

Assembler Präfix: *in_*

Datenstrukturenreferenz »Intuition«: *FatIntuiMessage*, *IntuitionBase*, *IntuiMessage*, *Window*.

Routinenreferenz »Exec«: *GetMsg*, *ReplyMsg*.

Beschreibung

Die *IntuiMessage*-Datenstruktur ist eine Erweiterung einer *ExecMessege*-Datenstruktur. Sie ist eine vollständige Systemnachricht zur Meldung von Ereignissen, die im Zusammenhang mit der »intuition.library« passieren, wie beispielsweise die Aktivierung eines Gadgets.

❑ **ExecMessage 0x0000 (0)**

Hier ist eine *ExecMessage*-Datenstruktur eingebunden.

❑ **Class 0x0014 (20)**

Hier stehen die *IDCMP-Flags* des nächsten eingetragenen Ereignisses zur Abfrage bereit. Eine genaue Beschreibung der *IDCMP-Flags* finden Sie im Abschnitt »Wichtige Flaggs der Intuition-Library«.

- **Code 0x0018 (24)** — Hier können Zusatzdaten, wie beispielsweise ASCII- oder Raw-Code einer gedrückten Taste stehen. Um welchen Parameter es sich dabei handelt, wird durch die *IDCMP-Flags* in *Class* bestimmt.
- **Qualifier 0x001A (26)** — Handelt es sich um eine Tastatureingabe, so kann diesem Wert entnommen werden, welche zusätzlichen Kontrolltasten (<Shift>, <Alt>, <Ctrl>, etc.) zur gleichen Zeit gedrückt waren. Die genauen Werte der einzelnen Kontrolltasten finden Sie in der Beschreibung der *InputEvent*-Datenstruktur.
- **IAddress 0x001C (28)** — Ein Adreßzeiger auf die Datenstruktur des zugehörigen Displayelements. Wurde beispielsweise ein Gadget aktiviert, so steht hier der Adreßzeiger auf die entsprechende *Gadget*-Datenstruktur.
- **MouseX 0x0020 (32)** — Die x-Koordinate des Mauszeigers, relativ zur linken oberen Ecke des zugehörigen Windows, zum Zeitpunkt des Ereignisses.
- **MouseY 0x0022 (34)** — Die y-Koordinate des Mauszeigers, relativ zur linken oberen Ecke des zugehörigen Windows, zum Zeitpunkt des Ereignisses.
- **Seconds 0x0024 (36)** — Sekundenanteil der Systemzeit zum Zeitpunkt des Ereignisses.
- **Micros 0x0028 (40)** — Mikrosekundenanteil (zwischen 0 und 999) der Systemzeit zum Zeitpunkt des Ereignisses.
- **IDCMPWindow 0x002C (44)** — Ein Adreßzeiger auf die *Window*-Datenstruktur des Windows, zu dem diese Nachricht gehört.
- **SpecialLink 0x0030 (48)** — Systeminterner Adreßzeiger auf weitere vorliegende *IntuiMessage*-Datenstrukturen in der Message-Liste.

IntuiText

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct IntuiText	IntuiText =
		{	RECORD
0000	0	UBYTE FrontPen;	frontPen:UByte;
0001	1	UBYTE BackPen;	backPen:UByte;
0002	2	UBYTE DrawMode;	drawMode:DrawModeSet;
0004	4	SHORT LeftEdge;	leftEdge:INTEGER;
0006	6	SHORT TopEdge;	topEdge:INTEGER;
0008	8	struct TextAttr *ITextFont;	iTextFont:TextAttrPtr;
000C	12	UBYTE *IText;	iText:ADDRESS;

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
0010	16	struct IntuiText *NextText;	nextText:IntuiTextPtr;
0014	20	};	END;

Assembler Präfix: *it_*

Datenstrukturenreferenz »Intuition«: *Gadget, IntuiText, Requester.*

Routinenreferenz »Intuition«: *AutoRequest, BuildSysRequest, PrintIText.*

4

Beschreibung

Die *IntuiText*-Datenstruktur definiert einen Text (samt seiner Eigenschaften wie Schriftart und -stil), der durch die *PrintIText*-Routine ausgegeben werden kann.

- **FrontPen 0x0000 (0)** Die Nummer des Farbregisters, dessen Inhalt beim Schreiben des Textes als Vordergrundfarbe verwendet wird.
- **BackPen 0x0001 (1)** Die Nummer des Farbregisters, dessen Inhalt beim Schreiben des Textes als Hintergrundfarbe verwendet wird.
- **DrawMode 0x0002 (2)** Der Zeichenmodus, mit dem der Text geschrieben wird. Zur Verfügung stehen *JAM1* (Schreiben des Textes mit der Vordergrundfarbe *FrontPen*), *JAM2* (zusätzliches Beschreiben der Umgebung der Buchstaben mit der Hintergrundfarbe *BackPen*) und *XOR* (Invertieren des Textes).
- **LeftEdge 0x0004 (4)** Die zu der x-Koordinate, die beim Aufruf in der *PrintIText*-Routine angegeben wird, relative x-Koordinate der oberen linken Ecke der Zeichenkette.
- **TopEdge 0x0006 (6)** Die zu der y-Koordinate, die beim Aufruf in der *PrintIText*-Routine angegeben wird, relative y-Koordinate der oberen linken Ecke der Zeichenkette.
- **ITextFont 0x0008 (8)** Ein Adreßzeiger auf eine *TextAttr*-Datenstruktur, die die Informationen über den zu verwendenden Zeichensatz enthält. Ist dieser Wert *NULL*, so wird der Default-Zeichensatz verwendet.
- **IText 0x000C (12)** Ein Adreßzeiger auf die auszugebende, Null-terminierte Zeichenkette.
- **NextText 0x0010 (16)** Ein Adreßzeiger auf weitere *IntuiText*-Datenstrukturen, die mit dem gleichen Aufruf auch noch ausgegeben werden sollen. Steht hier *NULL*, so war dies die letzte *IntuiText*-Datenstruktur.

4

IntuitionBase

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	Dez	struct IntuitionBase {	IntuitionBase = RECORD
0000	0	struct Library LibNode;	libNode:Library;
0022	34	struct View ViewLord;	viewLord:View;
0034	52	struct Window *ActiveWindow;	activeWindow:WindowPtr;
0038	56	struct Screen *ActiveScreen;	activeScreen:ScreenPtr;
003C	60	struct Screen *FirstScreen;	firstScreen:ScreenPtr;
0040	64	ULONG Flags;	flags:LONGSET;
0044	68	WORD MouseY;	mouseY:INTEGER;
0046	70	WORD MouseX;	mouseX:INTEGER;
0048	72	ULONG Seconds;	seconds:LONGCARD;
004C	76	ULONG Micros;	micros:LONGCARD;
0050	80	WORD MinXMouse;	minXMouse:INTEGER;
0052	82	WORD MaxXMouse;	maxXMouse:INTEGER;
0054	84	WORD MinYMouse;	minYMouse:INTEGER;
0056	86	WORD MaxYMouse;	maxYMouse:INTEGER;
0058	88	ULONG StartSecs;	startSecs:LONGCARD;
005C	92	ULONG StartMicros;	startMicros:LONGCARD;
0060	96	APTR SysBase;	sysBase:ADDRESS;
0064	100	struct GfxBase *GfxBase;	gfxBase:GfxBasePtr;
0068	104	APTR LayersBase;	layersBase:ADDRESS;
006C	108	APTR ConsoleDevice;	consoleDevice:ADDRESS;
0070	112	USHORT *APointer;	aPointer:ADDRESS;
0074	116	BYTE APtrHeight;	aPtrHeight:UByte;
0075	117	BYTE APtrWidth;	aPtrWidth:[0..16];
0076	118	BYTE AXOffset;	aXOffset:UByte;
0077	119	BYTE AYOffset;	aYOffset:UByte;
0078	120	USHORT MenuDrawn;	menuDrawn:CARDINAL;
007A	122	USHORT MenuSelected;	menuSelected:CARDINAL;
007C	124	USHORT OptionList;	optionList:CARDINAL;
007E	126	struct RastPort MenuRPort;	menuRPort:RastPort;
00E2	226	struct TmpRas MenuTmpRas;	menuTmpRas:TmpRas;
00EA	234	struct ClipRect ItemCRect;	itemCRect:ClipRect;
0112	274	struct ClipRect SubCRect;	subCRect:ClipRect;
013A	314	struct BitMap IBitMap;	iBitMap:BitMap;
0162	354	struct BitMap SBitMap;	sBitMap:BitMap;
018A	394	struct IOStdReq InputRequest;	inputRequest:IOStdReq;
01BA	442	struct Interrupt InputInterrupt;	inputInterrupt:Interrupt;
01C0	464	struct Remember *EventKey;	eventKey:RememberPtr;
01C4	468	struct InputEvent *IEvents;	iEvents:ADDRESS;
01C8	472	SHORT EventCount;	eventCount:INTEGER;

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	Dez		
01CA	474	struct InputEvent IEBuffer[4];	ieBuffer: ARRAY [0..3] OF InputEvent;
0232	562	struct Gadget *ActiveGadget;	activeGadget:GadgetPtr;
0236	566	struct PropInfo *ActivePInfo;	activePInfo:PropInfoPtr;
023A	570	struct Image *ActiveImage;	activeImage:ImagePtr;
023E	574	struct GListEnv GadgetEnv;	gadgetEnv:GListEnv;
0268	616	struct GadgetInfo GadgetInfo;	gadgetInfo:GadgetInfo;
028C	652	struct Point KnobOffset;	knobOffset:Point;
0290	656	struct Window *getOKWindow;	getOKWindow:WindowPtr;
0294	660	struct IntuiMessage *getOKMessage;	getOKMessage: IntuiMessagePtr;
0298	664	USHORT setWExcept;	setWExcept:CARDINAL;
029A	666	USHORT GadgetReturn;	gadgetReturn:CARDINAL;
029C	668	USHORT StateReturn;	stateReturn:CARDINAL;
029E	670	struct RastPort *RP;	rp:RastPortPtr;
02A2	674	struct TmpRas ITmpRas;	iTmpRas:TmpRas;
02AA	682	struct Region *OldClipRegion;	oldClipRegion:RegionPtr;
02AE	686	struct Point OldScroll;	oldScroll:Point;
02B2	690	struct IBox IFrame;	iFrame:IBox;
02BA	698	SHORT hthick;	hthick:INTEGER;
02BC	700	SHORT vthick;	vthick:INTEGER;
02BE	702	VOID (*frameChange)();	frameChange:PROC;
02C2	706	VOID (*sizeDrag)();	sizeDrag:PROC;
02C6	710	struct Point FirstPt;	firstPt:Point;
02CA	714	struct Point OldPt;	oldPt:Point;
02CE	718	struct Gadget *SysGadgets[16];	sysGadgets: ARRAY [0..15] OF GadgetPtr;
030E	782	struct Image *CheckImage[2];	checkImage: ARRAY [0..1] OF ImagePtr;
0316	790	struct Image *AmigaIcon[2];	amigaIcon: ARRAY [0..1] OF ImagePtr;
031E	798	USHORT apattern[8];	aPattern: ARRAY [0..7] OF CARDINAL;
032E	814	USHORT bpattern[4];	bPattern: ARRAY [0..3] OF CARDINAL;
0336	822	USHORT *IPointer;	iPointer:ADDRESS;
033A	826	BYTE IPtrHeight;	iPtrHeight:UByte;
033B	827	BYTE IPtrWidth;	iPtrWidth:[0..16];
033C	828	BYTE IXOffset;	iXOffset:UByte;
033D	829	BYTE IYOffset;	iYOffset:UByte;
033E	830	LONG DoubleSeconds;	doubleSeconds:LONGINT;
0342	834	LONG DoubleMicros;	doubleMicros:LONGINT;

4

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	Dez		
0346	838	BYTE WBorLeft[2];	wBorLeft,
0348	840	BYTE WBorTop[2];	wBorTop,
034A	842	BYTE WBorRight[2];	wBorRight,
034C	844	BYTE WBorBottom[2];	wBorBottom,
034E	846	BYTE BarVBorder[2];	barVBorder,
0350	848	BYTE BarHBorder[2];	barHBorder,
0352	850	BYTE MenuVBorder[2];	menuVBorder,
0354	852	BYTE MenuHBorder[2];	menuHBorder: ARRAY [0..1] OF Byte;
0356	854	USHORT color0;	color0:CARDINAL;
0358	856	USHORT color1;	color1:CARDINAL;
035A	858	USHORT color2;	color2:CARDINAL;
035C	860	USHORT color3;	color3:CARDINAL;
035E	862	USHORT color17;	color17:CARDINAL;
0360	864	USHORT color18;	color18:CARDINAL;
0362	866	USHORT color19;	color19:CARDINAL;
0364	868	struct TextAttr SysFont;	sysFont:TextAttr;
036C	876	struct Preferences *Preferences;	preferences:PreferencesPtr;
0370	880	struct DistantEcho *Echoes;	echoes:ADDRESS;
0374	884	WORD ViewInitX;	viewInitX:INTEGER;
0376	886	WORD ViewInitY;	viewInitY:INTEGER;
0378	888	SHORT CursorDX;	cursorDX:INTEGER;
037A	890	SHORT CursorDY;	cursorDY:INTEGER;
037C	892	struct KeyMap *KeyMap;	keyMap:KeyMapPtr;
0380	896	SHORT MouseYMinimum;	mouseYMinimum:INTEGER;
0382	898	SHORT ErrorX;	errorX:INTEGER;
0384	900	SHORT ErrorY;	errorY:INTEGER;
0386	902	struct timerequest IOExcess;	ioExcess:IOTimer;
03AE	942	SHORT HoldMinYMouse;	holdMinYMouse:INTEGER;
03B0	944	struct MsgPort *WBPort;	wbPort:MsgPortPtr;
03B4	948	struct MsgPort *iqd_FNKUHDPort;	fnkuhdPort:MsgPortPtr;
03B8	952	struct IntuiMessage WBMessage;	wbMessage:IntuiMessage;
03EC	1004	struct Screen *HitScreen;	hitScreen:ScreenPtr;
03F0	1008	struct SimpleSprite *SimpleSprite;	simpleSprite: SimpleSpritePtr;
03F4	1012	struct SimpleSprite *AttachedSSprite;	attachedSSprite: SimpleSpritePtr;
03F8	1016	BOOL GotSprite1;	gotSpritel:BOOLEAN;
03FA	1018	struct List SemaphoreList;	semaphoreList:List;
0408	1032	struct SignalSemaphore	iSemaphore:ARRAY [0..6] OF

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	Dez		
		ISemaphore[7];	SignalSemaphore;
054A	1354	WORD MaxDisplayHeight;	maxDisplayHeight:INTEGER;
054C	1356	WORD MaxDisplayRow;	maxDisplayRow:INTEGER;
054E	1358	WORD MaxDisplayWidth;	reserved:ARRAY [0..7] OF
0550	1360	ULONG Reserved[7];	LONGCARD;
056E	1390	};	END;

4

Assembler — Präfix: *ib_*
In Assembler sind nur die ersten fünf Komponenten dieser systeminternen Datenstruktur definiert.

Datenstrukturenreferenz — Keine.

Routinenreferenz — »Intuition«: *LockIBase*, *OpenIntuition*, *UnLockIBase*.

Beschreibung

Diese systeminterne Datenstruktur enthält einige Daten, die bei der Fehlersuche helfen können. Um zu verhindern, daß Werte verändert werden, während Sie sich diese ansehen, rufen Sie bitte vorher die *LockIBase*-Routine auf. (Den Adreßzeiger auf diese Datenstruktur erhalten Modula-2-Programmierer durch die Routine *OpenIntuition*.) Haben Sie genug gesehen, können Sie durch *UnLockIBase* die *IntuitionBase*-Datenstruktur wieder freigeben. Da es nicht gewährleistet ist, daß sich diese Parameter in zukünftigen Versionen nicht verändern, sollten sie von Programmen nicht benutzt, auf gar keinen Fall aber verändert werden.

❑ **LibNode 0x0000 (0)** — Ein Adreßzeiger auf die *Library*-Datenstruktur, die für die Einbindung der »intuition.library« in das System zuständig ist. Eine genaue Beschreibung dieser Datenstruktur finden Sie im Kapitel über »Exec«.

❑ **ViewLord 0x0022 (34)** — Einbindung einer *View*-Datenstruktur, die die Definition des Gesamtdisplays übernimmt.

❑ **ActiveWindow 0x0034 (52)** — Ein Adreßzeiger auf die *Window*-Datenstruktur des momentan aktiven Fensters.

❑ **ActiveScreen 0x0038 (56)** — Ein Adreßzeiger auf die *Screen*-Datenstruktur des momentan aktiven Bildschirms.

❑ **FirstScreen 0x003C (60)** — Ein Adreßzeiger auf die *Screen*-Datenstruktur des in Vordergrund stehenden Bildschirms.

❑ **Flags 0x0040 (64)** — Eine systeminterne Kombination von verschiedenen Flags der »intuition.library«.

❑ **MouseY 0x0044 (68)** — Die absolute y-Koordinate der augenblicklichen Position des Mauszeigers.

- **MouseX 0x0046 (70)** Die absolute y-Koordinate der augenblicklichen Position des Mauszeigers.
- **Seconds 0x0048 (72)** Der Sekundenanteil der Systemzeit zum Zeitpunkt des aktuellsten Ereignisses (*Inputevent*).
- **Micros 0x004C (76)** Der Mikrosekundenanteil der Systemzeit zum Zeitpunkt des aktuellsten Ereignisses (*Inputevent*).

ACHTUNG: Alle folgenden Parameter dienen nur zur Fehlersuche! Ein Verändern der folgenden Werte kann nach einer ungewissen Anzahl von Taktzyklen einen nicht wünschenswerten Systemzustand heraufbeschwören.

- **MinXMouse 0x0050 (80)** Linker Begrenzungswert für den Bereich, in dem der Benutzer den Mauszeiger bewegen kann.
- **MaxXMouse 0x0052 (82)** Rechter Begrenzungswert für den Bereich, in dem der Benutzer den Mauszeiger bewegen kann.
- **MinYMouse 0x0054 (84)** Oberer Begrenzungswert für den Bereich, in dem der Benutzer den Mauszeiger bewegen kann.
- **MaxYMouse 0x0056 (86)** Unterer Begrenzungswert für den Bereich, in dem der Benutzer den Mauszeiger bewegen kann.
- **StartSecs 0x0058 (88)** Der Sekundenanteil der Zeitdauer, die zwischen zwei »Maus-Klicks« liegen darf, damit dieser Vorgang als »Doppel-Klick« erkannt wird.
- **StartMicros 0x005C (92)** Der Mikrosekundenanteil der Zeitdauer, die zwischen zwei »Maus-Klicks« liegen darf, damit dieser Vorgang als »Doppel-Klick« erkannt wird.
- **SysBase 0x0060 (96)** Ein Adreßzeiger auf die Basisadresse des Systems, also auf die entsprechende *Exec*-Datenstruktur.
- **GfxBase 0x0064 (100)** Ein Adreßzeiger auf die aktuelle *GfxBase*-Datenstruktur.
- **LayersBase 0x0068 (104)** Ein Adreßzeiger auf die aktuelle *LayersBase*-Datenstruktur.
- **ConsoleDevice 0x006C (108)** Ein Adreßzeiger auf das aktuelle Console-Device.
- **APointer 0x0070 (112)** Ein Adreßzeiger auf die Grafikdaten des momentan aktiven Mauszeigers.
- **APtrHeight 0x0074 (116)** Die Höhe des momentan aktiven Mauszeigers.
- **APtrWidth 0x0075 (117)** Die Breite des momentan aktiven Mauszeigers.
- **AXOffset 0x0076 (118)** Der Abstand des Hot-Spot vom linken Rand des momentan aktiven Mauszeigers.
- **AYOffset 0x0077 (119)** Der Abstand des Hot-Spot vom oberen Rand des momentan aktiven Mauszeigers.

- **MenuDrawn 0x0078 (120)** Die Menünummer des momentan gezeichneten Pull-down-Menüs. Wird zur Zeit kein Pull-down-Menü ausgegeben, so steht hier 0xFFFF.
- **MenuSelected 0x007A (122)** Die Menünummer des momentan ausgewählten Menüpunktes. Ist zur Zeit kein Menüpunkt ausgewählt, so steht hier 0xFFFF.
- **OptionList 0x007C (124)** Zusatzwert für die Options-Liste der Pull-down-Menüs.

BEACHTE: Die folgenden sechs Parameter beinhalten Informationen über den Rastport, der zur Darstellung der momentanen Menüs benutzt wird. Dabei handelt es sich im Normalfall um Kopien der entsprechenden Werte des augenblicklich aktiven Screens.

4

- **MenuRPort 0x007E (126)** Eine eingebundene *RastPort*-Datenstruktur, die beim Zeichnen der Menüs benutzt wird.
- **MenuTmpRas 0x00E2 (226)** Eine eingebundene *TmpRas*-Datenstruktur, die zu *MenuRPort* gehört.
- **ItemCRect 0x00EA (234)** Eine eingebundene *ClipRect*-Datenstruktur, die bei der Ausgabe von Pull-down-Menüs (Menuitems) verwendet wird.
- **SubCRect 0x0112 (274)** Eine eingebundene *ClipRect*-Datenstruktur, die bei der Ausgabe von Untermenüs (Menusubitems) verwendet wird.
- **IBitMap 0x013A (314)** Eine eingebundene *BitMap*-Datenstruktur, die bei der Ausgabe von Pull-down-Menüs (Menuitems) verwendet wird.
- **SBitMap 0x0162 (354)** Eine eingebundene *BitMap*-Datenstruktur, die bei der Ausgabe von Untermenüs (Menusubitems) verwendet wird.
- **InputRequest 0x018A (394)** Eine eingebundene *IOStdReq*-Datenstruktur, die für das aktuelle Input-Device gültig ist.
- **InputInterrupt 0x01BA (442)** Eine eingebundene *Interrupt*-Datenstruktur, die für das aktuelle Input-Device gültig ist.
- **EventKey 0x01C0 (464)** Ein Adreßzeiger auf eine *Remember*-Datenstruktur, die für dynamisch angelegte Input-Events verwendet wird.
- **IEvents 0x01C4 (468)** Ein Adreßzeiger auf die zu *EventKey* gehörende *InputEvent*-Datenstruktur.
- **EventCount 0x01C8 (472)** Ein Zähler für fest angelegte Input-Events.
- **IEBuffer 0x01CA (474)** Eine Einbindung von vier *InputEvent*-Datenstrukturen, die für fest angelegte Input-Events verwendet werden.
- **ActiveGadget 0x0232 (562)** Ein Adreßzeiger auf die *Gadget*-Datenstruktur des momentan aktivierten Gadgets.
- **ActivePInfo 0x0236 (566)** Handelt es sich bei dem momentan aktivierten Gadget um ein Proportional-Gadget, so steht hier ein Adreßzeiger auf die zugehörige *PropInfo*-Datenstruktur.

- **ActiveImage 0x023A (570)** Wird zu dem momentan aktivierten Gadget gerade ein Image ausgegeben, so steht hier ein Adreßzeiger auf die zugehörige *Image*-Datenstruktur.
- **GadgetEnv 0x023E (574)** Eine eingebundene *GListEnv*-Datenstruktur, in der nähere Informationen über die Umgebung des momentan aktiven Gadgets abgelegt werden.
- **GadgetInfo 0x0268 (616)** Eine eingebundene *GadgetInfo*-Datenstruktur, in der weitere Informationen über momentan aktive Gadgets abgelegt werden.
- **KnobOffset 0x028C (652)** Wurde ein Proportional-Gadget aktiviert, so steht in dieser eingebundenen *Point*-Datenstruktur die Position des Mauszeigers (zum Zeitpunkt der Aktivierung) innerhalb des Knopfes.
- **GetOKWindow 0x0290 (656)** Laut [KRA] gehören die *GetOk*xxx-Komponenten zu den »verify functions«. Eine durchaus zulässige Interpretation ist, daß damit die Intuition-Funktionen gemeint sind, die zu den IDCMP-Flags *SIZEVERIFY* und *REQVERIFY* oder zu dem Gadget-Flag *RELVERIFY* gehören.
- **GetOKMessage 0x0294 (660)** Ein Adreßzeiger auf eine *IntuiMessage*-Datenstruktur (siehe auch *GetOkWindow*).
- **SetWExcept 0x0298 (664)** Die einzige auftreibbare Dokumentation [KRA] zu den nächsten drei Komponenten besagt, daß sie sich auf »State Machine« beziehen. Was damit gemeint sein könnte, überlassen wir Ihrer persönlichen Intuition.
- **GadgetReturn 0x029A (666)** (siehe *SetWExept*)
- **StateReturn 0x029C (668)** (siehe *SetWExept*)
- **RP 0x029E (670)** Ein Adreßzeiger auf die *RastPort*-Datenstruktur, die bei der Ausgabe von Gadgets, Titelleisten, etc. verwendet wird.
- **ITmpRas 0x02A2 (674)** Einbindung einer zu *RP* gehörenden *TmpRas*-Datenstruktur.
- **OldClipRegion 0x02AA (682)** Ein Adreßzeiger auf eine zu *RP* gehörenden *Region*-Datenstruktur.
- **OldScroll 0x02AE (686)** Eine eingebundene *Point*-Datenstruktur, in der bei Verschiebungen die alte Position zwischengespeichert wird.
- **IFrame 0x02B2 (690)** Eine eingebundene *IBox*-Datenstruktur, in der beim Verschieben, Verkleinern oder Vergrößern von Fenstern die vorherige Position und Größe des Fensters zwischengespeichert wird.
- **HtHick 0x02BA (698)** Die Höhe der horizontalen Fensterränder.
- **VtHick 0x02BC (700)** Die Breite der vertikalen Fensterränder.

- **FrameChange 0x02BE (702)** Ein Adreßzeiger auf eine systeminterne Routine, die *IFrame* verändern kann.

- **SizeDrag 0x02C2 (706)** Ein Adreßzeiger auf eine von zwei systeminternen Routinen, je nachdem, ob die Größe des Fensters verändert oder es verschoben wird.

- **FirstPt 0x02C6 (710)** Eine eingebundene *Point*-Datenstruktur, in der bei Größenveränderungen von Fenstern die ursprüngliche Position der unteren rechten Ecke, bei Verschiebungen die der oberen linken Ecke zwischengespeichert wird.

- **OldPt 0x02CA (714)** Eine eingebundene *Point*-Datenstruktur, in der bei Größenveränderungen von Fenstern die zuletzt angezeigte Position der unteren rechten Ecke, bei Verschiebungen die der oberen linken Ecke zwischengespeichert wird.

- **SysGadgets 0x02CE (718)** An dieser Stelle stehen 16 Adreßzeiger auf verschiedene *Gadget*-Datenstrukturen, die die acht verschiedenen Systemgadgets für den HiRes- und LoRes- Modus definieren. Die Reihenfolge der Systemgadgets ist dabei: *WUPFRONT*, *WDOWNBACK*, *SIZING*, *CLOSE*, *WDRAGGING*, *SUPFRONT*, *SDOWNBACK* und *SDRAGGING*. Eine genauere Beschreibung finden Sie bei der Beschreibung von *GadgetType* der *Gadget*-Datenstruktur.

- **CheckImage 0x030E (782)** Zwei Adreßzeiger auf die Image-Datenstrukturen, die für die Modi *HiRes* und *LoRes* die Checkmark-Grafiksymbole definieren.

- **AmigaIcon 0x0316 (790)** Zwei Adreßzeiger auf die *Image*-Datenstrukturen, die für die Modi *HiRes* und *LoRes* die Grafiksymbole für die Amigataste (zur Darstellung in den Menüs) definieren.

- **APattern 0x031E (798)** Hier stehen acht Muster (die etwas mit dem Verschieben von Fenstern zu tun haben könnten ???).

- **BPattern 0x032E (814)** Hier stehen vier Muster (die auch etwas mit dem Verschieben von Fenstern zu tun haben könnten ???).

- **IPointer 0x0336 (822)** Ein Adreßzeiger auf die Grafikdaten des Default-Mauszeigers.

- **IPtrHeight 0x033A (826)** Die Höhe des Default-Mauszeigers.

- **IPtrWidth 0x033B (827)** Die Breite des Default-Mauszeigers (muß kleiner oder gleich 16 sein).

- **IXOffset 0x033C (828)** Der Abstand des Hot-Spot vom linken Rand des Default-Mauszeigers.

- **IYOffset 0x033D (829)** Der Abstand des Hot-Spot vom oberen Rand des Default-Mauszeigers.

- **DoubleSeconds 0x033E (830)** Der Sekundenanteil der Zeitdauer, die beim Default-Mauszeiger zwischen zwei »Maus-Klicks« liegen darf, damit dieser Vorgang als »Doppel-Klick« erkannt wird.
- **DoubleMicros 0x0342 (834)** Der Mikrosekundenanteil der Zeitdauer, die beim Default-Mauszeiger zwischen zwei »Maus-Klicks« liegen darf, damit dieser Vorgang als »Doppel-Klick« erkannt wird.

ACHTUNG: Die folgenden acht Werte sind jeweils doppelt vorhanden. Der erste gilt für den *HiRes*-, der zweite für den *LoRes*-Modus.

- **WBorLeft 0x0346 (838)** Die Breite der linken Fensterränder. Dieser Wert gilt für alle Fenster, die auf dem Workbench-Screen geöffnet werden.
- **WBorTop 0x0348 (840)** Die Höhe der oberen Fensterränder. Dieser Wert gilt für alle Fenster, die auf dem Workbench-Screen geöffnet werden.
- **WBorRight 0x034A (842)** Die Breite der rechten Fensterränder. Dieser Wert gilt für alle Fenster, die auf dem Workbench-Screen geöffnet werden.
- **WBorBottom 0x034C (844)** Die Höhe der unteren Fensterränder. Dieser Wert gilt für alle Fenster, die auf dem Workbench-Screen geöffnet werden.
- **BarVBorder 0x034E (846)** Die Höhe der vertikalen Bildschirmränder.
- **BarHBorder 0x0350 (848)** Die Breite der horizontalen Bildschirmränder.
- **MenuVBorder 0x0352 (850)** Die Höhe der vertikalen Ränder der Pull-down-Menüs.
- **MenuHBorder 0x0354 (852)** Die Breite der horizontalen Ränder der Pull-down-Menüs.
- **Color0 0x0356 (854)** Der Farbwert, der als Hintergrundfarbe verwendet wird.
- **Color1 0x0358 (856)** Der Farbwert, der als Vordergrundfarbe (z.B. Text und Windows) verwendet wird.
- **Color2 0x035A (858)** Der Farbwert, der als dritte Farbe (z.B. in den Depth-Gadgets) verwendet wird.
- **Color3 0x035C (860)** Der Farbwert, der als vierte Farbe (z.B. für den Textcursor) verwendet wird.
- **Color17 0x035E (862)** Der Farbwert, der als erste Farbe für die Darstellung des Mauszeigers verwendet wird.
- **Color18 0x0360 (864)** Der Farbwert, der als zweite Farbe für die Darstellung des Mauszeigers verwendet wird.
- **Color19 0x0362 (866)** Der Farbwert, der als dritte Farbe für die Darstellung des Mauszeigers verwendet wird.
- **SysFont 0x0364 (868)** Einbindung einer *TextAttr*-Datenstruktur, die den Systemzeichensatz definiert.
- **Preferences 0x036C (876)** Ein Adreßzeiger auf die momentan gültige *Preferences*-Datenstruktur.

- **Echoes 0x0370 (880)** — Ein Adreßzeiger auf eine sehr systeminterne Datenstruktur (wir haben sie noch nicht gefunden).
- **ViewInitX 0x0374 (884)** — Der x-Initialisierungswert für alle Views.
- **ViewInitY 0x0376 (886)** — Der y-Initialisierungswert für alle Views.
- **CursorDX 0x0378 (888)** — Dieser Wert gibt den Übersetzungsfaktor der Mausbewegungen in x-Richtung an. Siehe auch unter *PointerTicks* der *Preferences*-Datenstruktur.
- **CursorDY 0x037A (890)** — Dieser Wert gibt den Übersetzungsfaktor der Mausbewegungen in y-Richtung an. Siehe auch unter *PointerTicks* der *Preferences*-Datenstruktur.
- **KeyMap 0x037C (892)** — Ein Adreßzeiger auf die *KeyMap*-Datenstruktur, die bei Tastatureingaben in String-Gadgets verwendet wird.
- **MouseYMinimum 0x0380 (896)** — Die einzige auftreibbare Dokumentation [KRA] zu diesem Wert lautet »magic«. Also lieber die Finger davon lassen . . .
- **ErrorX 0x0382 (898)** — Wird zur Verfolgung der Mausbewegung in x-Richtung benutzt – aber wie?
- **ErrorY 0x0384 (900)** — Wird zur Verfolgung der Mausbewegung in y-Richtung benutzt – aber wie?
- **IOExcess 0x0386 (902)** — Eine eingebundene *TimeRequest*-Datenstruktur, mit deren Hilfe »Intuition« auf den Timer-Device zugreifen kann.
- **HoldMinYMouse 0x0386 (902)** — Im Moment spricht nichts dagegen, daß dieser Wert zu *MouseYMinimum* gehören könnte.
- **WBPort 0x03B0 (944)** — Ein Adreßzeiger auf den Messageport der Workbench.
- **iqd_FNKUHDPort 0x03B4 (948)** — Ein Adreßzeiger auf den Messageport des »Final Nuclear Knockout Unit Handling Device« (zumindest so lange, bis uns jemand etwas anderes beweist) ???
- **WBMessage 0x03B8 (952)** — Eine eingebundene *IntuiMessage*-Datenstruktur, für die Nachrichten von und zum Workbench-Fenster.
- **HitScreen 0x03EC (1004)** — Ein Adreßzeiger auf eine *Screen*-Datenstruktur. Dieser wird verwendet, wenn ein Screen in den Vordergrund geholt wird.
- **SimpleSprite 0x03F0 (1008)** — Ein Adreßzeiger auf eine *SimpleSprite*-Datenstruktur, deren Bedeutung hier nicht ganz klar ist.
- **AttachedSSprite 0x03F4 (1012)** — (Siehe *SimpleSprite*)
- **GotSprite1 0x03F8 (1016)** — Ein Boolescher Ausdruck, der wahrscheinlich in einem bestimmten Zusammenhang mit *SimpleSprite* und *AttachedSprite* steht.
- **SemaphoreList 0x03FA (1018)** — Eine eingebundene *List*-Datenstruktur, die zu *ISemaphore* einen Header bildet.

- **ISemaphore 0x0408 (1032)** Ein Vorrat von sieben *SignalSemaphore*-Datenstrukturen, die für interne Zwecke bereitgehalten werden.
- **MaxDisplayHeight 0x054A (1354)** Hier steht im Interlace-Modus die Anzahl der Bildzeilen, die dem System zur Verfügung stehen (400 oder 512).
- **MaxDisplayRow 0x054C (1356)** Hier steht der Wert aus *MaxDisplayHeight,* um eins dekrementiert.
- **MaxDisplayWidth 0x054E (1358)** Hier steht eine Kopie des Wertes *NormalDisplayCol* aus der *GFXBase*-Datenstruktur.
- **Reserved 0x0550 (1360)** Reservierter Bereich für zukünftige Erweiterungen.

Menu

Offset Hex	Dez	C (intuition/intuition.h)	Modula-2 (Intuition)
		struct Menu	Menu =
		{	RECORD
0000	0	struct Menu *NextMenu;	nextMenu:MenuPtr;
0004	4	SHORT LeftEdge;	leftEdge:INTEGER;
0006	6	SHORT TopEdge;	topEdge:INTEGER;
0008	8	SHORT Width;	width:INTEGER;
000A	10	SHORT Height;	height:INTEGER;
000C	12	USHORT Flags;	flags:BITSET;
000E	14	BYTE *MenuName;	menuName:ADDRESS;
0012	18	struct MenuItem *FirstItem;	firstItem:MenuItemPtr;
0016	22	SHORT JazzX;	jazzX:INTEGER;
0018	24	SHORT JazzY;	jazzY:INTEGER;
001A	26	SHORT BeatX;	beatX:INTEGER;
001C	28	SHORT BeatY;	beatY:INTEGER;
001E	30	};	END;

Assembler Präfix: *mu_*

Datenstrukturenreferenz »Intuition«: *Menu, Window.*

Routinenreferenz »Intuition«: *ClearMenuStrip, ItemAdress, OffMenu, OnMenu, SetMenuStrip.*

Beschreibung

In einer linearen Liste, bestehend aus miteinander verketteten *Menu*-Datenstrukturen, werden die einzelnen Menüpunkte der Menütitelleiste zu einem Window definiert (wie zum Beispiel die Menüpunkte »Workbench«, »Disk« und »Special« der Workbench). Die dazugehörigen Pull-down-Menüs werden jedoch durch *MenuItem*-Datenstrukturen definiert.

❑ **NextMenu 0x0000 (0)**

Ein Adreßzeiger auf die nächste *Menu*-Datenstruktur in der linearen Liste von angelegten Menüpunkten. Ist der Wert *NULL*, so ist dies das Ende der Liste.

❑ **LeftEdge 0x0004 (4)**

Die zum linken Bildschirmrand relative x-Koordinate der linken, oberen Ecke des zu diesem Menütitel gehörigen Auswahlkastens in der Menütitelleiste.

❑ **TopEdge 0x0006 (6)**

Die zum oberen Bildschirmrand relative y-Koordinate der linken, oberen Ecke des zu diesem Menütitel gehörigen Auswahlkastens in der Menütitelleiste. In den vorliegenden Versionen wird dieser Parameter von der »intuition.library« noch nicht beachtet. Es wird statt dessen die y-Koordinate des oberen Bildschirmrandes als aktueller Wert benutzt.

❑ **Width 0x0008 (8)**

Die Breite des zu diesem Menütitel gehörigen Auswahlkastens in der Menütitelleiste.

❑ **Height 0x000A (10)**

Die Breite und die Höhe des zu diesem Menütitel gehörigen Auswahlkastens in der Menütitelleiste. In den vorliegenden Versionen wird der Parameter Height von »Intuition« noch nicht beachtet. Es wird statt dessen die Höhe der Bildschirmtitelleiste als aktueller Wert benutzt.

❑ **Flags 0x000C (12)**

Eine 2-Byte-Marke, bei der folgende gesetzte Bits eine Bedeutung haben:

C	Modula-2	Hex	Beschreibung
MENUENABLED	menuEnabled	0001	Dieses Pull-down-Menü ist aktiviert.
MIDRAWN	miDrawn	0100	Dieses Pull-down-Menü wird momentan auf der Titelleiste dargestellt.

❑ **MenuName 0x000E (14)**

Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, die den Titel dieses Menüpunktes enthält.

❑ **FirstItem 0x0012 (18)**

Ein Adreßzeiger auf die erste von mehreren miteinander linear verketteten *MenuItem*-Datenstrukturen, die zu diesem Menüpunkt gehören.

❑ **JazzX 0x0016 (22)**

Von »Intuition« intern verwendeter Systemparameter.

❑ **JazzY 0x0018 (24)**

Von »Intuition« intern verwendeter Systemparameter.

❑ **BeatX 0x001A (26)**

Von »Intuition« intern verwendeter Systemparameter.

❑ **BeatY 0x001C (28)**

Von »Intuition« intern verwendeter Systemparameter.

4

MenuItem

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct MenuItem	MenuItem =
		{	RECORD
0000	0	struct MenuItem *NextItem;	nextItem:MenuItemPtr;
0004	4	SHORT LeftEdge;	leftEdge:INTEGER;
0006	6	SHORT TopEdge;	topEdge:INTEGER;
0008	8	SHORT Width;	width:INTEGER;
000A	10	SHORT Height;	height:INTEGER;
000C	12	USHORT Flags;	flags:MenuItemFlagSet;
000E	14	LONG MutualExclude;	mutualExclude:LONGSET;
0012	18	APTR ItemFill;	itemFill:ADDRESS;
0016	22	APTR SelectFill;	selectFill:ADDRESS;
001A	26	BYTE Command;	command:CHAR;
001C	28	struct MenuItem *SubItem;	subItem:MenuItemPtr;
0020	32	USHORT NextSelect;	nextSelect:CARDINAL;
0022	34	};	END;

Assembler Präfix: *mi_*

Datenstrukturenreferenz »Intuition«: *Menu*, *MenuItem*.

Routinenreferenz »Intuition«: *ItemAddress*, *OffMenu*, *OnMenu*.

Beschreibung

In einer linearen Liste, bestehend aus verketteten *MenuItem*-Datenstrukturen, werden alle Menüpunkte eines einzelnen Pull-down-Menüs definiert (wie zum Beispiel die Menüpunkte »Empty Trash« und »Initialize« im Pull-down-Menü »Disk« der Workbench). Dabei gibt es Action-Menüpunkte, bei denen einmalig etwas ausgeführt wird (z.B. Laden einer Datei) und *Attribute*-Menüpunkte, bei denen Zustände ein- und ausgeschaltet werden können (z.B. eine besondere Schriftart). Die zu dieser Datenstruktur gehörigen Basismenüpunkte der Menütitelleiste werden durch *Menu*-Datenstrukturen definiert.

❑ **NextItem 0x0000 (0)** Ein Adreßzeiger auf die nächste *MenuItem*-Datenstruktur in der linearen Liste von angelegten Menüpunkten. Ist der Wert *NULL*, so ist dies das Ende der Liste.

❑ **LeftEdge 0x0004 (4)** Die y-Koordinate der linken, oberen Ecke des Auswahlkastens zu diesem Menüpunkt, relativ zu dem *LeftEdge*-Wert in der zugehörigen *Menu*-Datenstruktur.

❑ **TopEdge 0x0006 (6)** Die x-Koordinate der linken, oberen Ecke des Auswahlkastens zu diesem Menüpunkt, relativ zu dem *TopEdge*-Wert in der zugehörigen *Menu*-Datenstruktur.

❑ **Width 0x0008 (8)** Die Breite des zu diesem Menütitel gehörigen Auswahlkastens in der Menütitelleiste.

- **Height 0x000A (10)**

Die Höhe des zu diesem Menütitel gehörigen Auswahlkastens in der Menütitelleiste.

- **Flags 0x000C (12)**

Eine 2-Byte-Marke mit Informationen über die Art und den Auswahlzustand dieses Menüpunktes. Ein gesetztes Bit hat jeweils die folgende Bedeutung:

C	Modula-2	Hex	Beschreibung
CHECKIT	checkIt	0001	Bei gesetztem Bit ist dies ein Attribute-, sonst ein Action-Menüpunkt.
ITEMTEXT	itemText	0002	Bei gesetztem Bit wird dieser Menüpunkt mit einem Text, sonst mit einem grafischen Symbol dargestellt.
COMMSEQ	commSeq	0004	Dieser Menüpunkt kann auch abgekürzt über die Tastatur aktiviert werden.
MENUTOGGLE	menuToggle	0008	Dies ist ein Attribute-Menüpunkt.
ITEMENABLED	itemEnabled	0010	Dieser Menüpunkt kann ausgewählt werden.
HIGHCOMP	highComp	0040	Bei Auswahl wird dieser Menüpunkt invertiert.
HIGHBOX	highBox	0080	Bei Auswahl wird dieser Menüpunkt umrandet.
CHECKED	checked	0100	Ist dies ein Attribute-Menüpunkt, so kann er bei gelöschtem Bit noch ausgewählt werden.
ISDRAWN	isDrawn	1000	Es wird gerade ein Untermenü angezeigt.
HIGHITEM	highItem	2000	Dieser Menüpunkt ist gerade ausgewählt.
MENUTOGGLED	menuToggled	4000	Dieser Attribute-Menüpunkt ist eingeschaltet.

- **MutualExclude 0x000E (14)**

Bei *Attribute*-Menüpunkten gibt das Bitmuster dieser 4-Byte-Maske an, ob andere *Attribute*-Menüpunkte dieses Untermenüs noch zugänglich sind, wenn dieser Menüpunkt bereits aktiviert ist. In einem Pull-down-Menü zur Wahl einer Schriftart ist es damit möglich, eine Kombination von bestimmten Schriftarten (wie hoch- und tiefgestellt) zu verbieten. Ein gelöschtes Bit bedeutet, daß der korrespondierende Menüpunkt dieses Pull-down-Menüs nicht mehr eingeschaltet werden kann. Entsprechend wird bei diesen Menüpunkten das *CHECKED*-Bit beeinflußt.

- **ItemFill 0x0012 (18)**

Dies ist ein Adreßzeiger auf eine *Image*- oder *IntuiText*-Datenstruktur, je nachdem, ob in dem Auswahlkasten eine

Grafik oder ein Text ausgegeben wird. Bestimmt wird dies durch das *ITEMTEXT*-Bit.

❑ **SelectFill 0x0016 (22)**

Dieser Adreßzeiger wird nur benötigt, wenn das *HIGH-ITEM*-Bit in *Flags* gesetzt ist. In diesem Fall zeigt dieser Adreßzeiger auf die alternative *Image-* oder *IntuiText*-Datenstruktur, die angeben, welche Grafik oder welcher Text bei der Auswahl dieses Menüpunktes ausgegeben wird.

❑ **Command 0x001A (26)**

Hier steht das alphanumerisches Zeichen, mit dem dieser Menüpunkt auch über die Tastatur (in Kombination mit der rechten Amiga-Taste) ausgewählt werden kann. Dieses Abkürzen funktioniert jedoch nur, wenn das *COMMSEQ*-Bit in *Flags* gesetzt ist.

❑ **SubItem 0x001C (28)**

Dies ist ein Adreßzeiger auf eine weitere Kette von *Menu-Item*-Datenstruktur, durch die ein weiteres Untermenü definiert werden kann. Ist kein solches Untermenü vorhanden, so steht hier Null.

❑ **NextSelect 0x0020 (32)**

In dieses Feld wird durch »Intuition« automatisch die Menünummer des nächsten, ausgewählten Menüpunktes eingetragen. Ist dieser Wert 0xFFFF (–1), so bedeutet dies, daß kein weiterer Menüpunkt ausgewählt ist.

NewScreen

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct NewScreen	NewScreen =
		{	RECORD
0000	0	SHORT LeftEdge;	leftEdge:INTEGER;
0002	2	SHORT TopEdge;	topEdge:INTEGER;
0004	4	SHORT Width;	width:INTEGER;
0006	6	SHORT Height;	height:INTEGER;
0008	8	SHORT Depth;	depth:INTEGER;
000A	10	UBYTE DetailPen;	detailPen:Byte;
000B	11	UBYTE BlockPen;	blockPen:Byte;
000C	12	USHORT ViewModes;	viewModes:ViewModeSet;
000E	14	USHORT Type;	type:ScreenFlagSet;
0010	16	struct TextAttr *Font;	font:TextAttrPtr;
0014	20	UBYTE *DefaultTitle;	defaultTitle:ADDRESS;
0018	24	struct Gadget *Gadgets;	gadgets:GadgetPtr;
001C	28	struct BitMap *CustomBitMap;	customBitMap:BitMapPtr;
0020	32	};	END;

Assembler Präfix: *ns_*

Datenstrukturenreferenz Keine.

Routinenreferenz »Intuition«: *OpenScreen.*

Beschreibung

Die *NewScreen*-Datenstruktur wird verwendet, um die Eigenschaften des Screens zu bestimmen, der durch die Routine *OpenScreen* geöffnet wird. Nach dem Öffnen gehört zu jedem Bildschirm eine *Screen*-Datenstruktur, die die jeweils aktuellen Parameter beinhaltet.

4

- **LeftEdge 0x0000 (0)** Die x-Koordinate der linken obere Ecke des Bildschirms. Stellen Sie bitte sicher, daß dieser Wert immer Null ist, da ein horizontales Verschieben in den vorliegenden Versionen noch nicht implementiert ist.
- **TopEdge 0x0002** Die y-Koordinate der linken oberen Ecke des Bildschirms.
- **Width 0x0004 (4)** Die Breite des Bildschirms, wobei dieser Wert nur 320 oder 640 sein darf, je nachdem, ob der Darstellungsmodus *LoRes* oder *HiRes* ist.
- **Height 0x0006 (6)** Die Höhe des Bildschirms, wobei die Summe aus diesem Wert und *TopEdge* im Interlace-Modus 512, sonst 256 ergeben muß.
- **Depth 0x0008 (8)** Die Anzahl der BitPlanes, die zu diesem Bildschirm gehören. Also je nach Darstellungsmodus zwischen 1 und 6.
- **DetailPen 0x000A (10)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für die »Details« verwendet wird, also z.B. für die Einzelheiten in den »Gadgets« und der Schrift in der Titelleiste.
- **BlockPen 0x000B (11)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für »gröbere« Elemente verwendet wird, also z.B. für das Füllen der Titelleiste.
- **ViewModes 0x000C (12)** Diese 2-Byte-Marke enthält die *ViewMode*-Flags, die unter anderem die Auflösung bestimmen. Eine genaue Beschreibung dieser Flags finden Sie im Kapitel über »Graphics«.
- **Type 0x000E (14)** Diese 2-Byte-Marke enthält die *Screen*-Flags. Eine genaue Beschreibung dieser Flags finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.
- **Font 0x0010 (16)** Ein Adreßzeiger auf eine *TextAttr*-Datenstruktur. Die in ihr eingestellten Parameter definieren für diesen Bildschirm einen Default-Font (wird in allen Displayelementen, wie z.B. Fenster, als voreingestellte Schriftart verwendet). Steht hier *NULL*, so wird der in der *IntuitionBase*-Datenstruktur voreingestellte Zeichensatz benutzt.

- **DefaultTitle 0x0014 (20)** Adreßzeiger auf eine Null-terminierte Zeichenkette, die als Überschrift in der Titelleiste des Bildschirms erscheint. Steht hier der Wert *NULL*, so wird kein Text ausgegeben.
- **Gadgets 0x0018 (24)** Dieses Feld muß *NULL* sein, da es in den vorliegenden Versionen noch keine Verwendung hat.
- **CustomBitMap 0x001C (28)** Wollen Sie den Displayspeicher zu diesem Bildschirm selbst bestimmen, so setzen Sie an diese Stelle einen Adreßzeiger auf eine von Ihnen vorbereitete *BitMap*-Datenstruktur. Zusätzlich muß dann das *CUSTOMBITMAP*-Bit in Types gesetzt werden.

NewWindow

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct NewWindow	NewWindow =
		{	RECORD
0000	0	SHORT LeftEdge;	leftEdge:INTEGER;
0002	2	SHORT TopEdge;	topEdge:INTEGER;
0004	4	SHORT Width;	width:INTEGER;
0006	6	SHORT Height;	height:INTEGER;
0008	8	UBYTE DetailPen;	detailPen:Byte;
0009	9	UBYTE BlockPen;	blockPen:Byte;
000A	10	ULONG IDCMPFlags;	idcmpFlags:IDCMPFlagSet;
000E	14	ULONG Flags;	flags:WindowFlagSet;
0012	18	struct Gadget *FirstGadget;	firstGadget:GadgetPtr;
0016	22	struct Image *CheckMark;	checkMark:ImagePtr;
001A	26	UBYTE *Title;	title:ADDRESS;
001E	30	struct Screen *Screen;	screen:ScreenPtr;
0022	34	struct BitMap *BitMap;	bitMap:BitMapPtr;
0026	38	SHORT MinWidth;	minWidth:INTEGER;
0028	40	SHORT MinHeight;	minHeight:INTEGER;
002A	42	USHORT MaxWidth;	maxWidth:INTEGER;
002C	44	USHORT MaxHeight;	maxHeight:INTEGER;
002E	46	USHORT Type;	type:ScreenFlagSet;
0030	48	};	END;

Assembler Präfix: *nw_*

Datenstrukturenreferenz »Workbench«: *DrawerData.*

Routinenreferenz »Intuition«: *OpenWindow.*

Beschreibung

Die *NewWindow*-Datenstruktur wird verwendet, um die Eigenschaften des Fensters festzulegen, das durch die Routine *OpenWindow* geöffnet wird. Nach dem Öffnen gehört zu jedem Fenster eine *Window*-Datenstruktur, die die jeweils aktuellen Parameter enthält.

- **LeftEdge 0x0000 (0)** — Der Abstand des Fensters vom linken Rand des zugehörigen Bildschirms.
- **TopEdge 0x0002 (2)** — Der Abstand des Fensters vom oberen Rand des zugehörigen Bildschirms.
- **Width 0x0004 (4)** — Die Breite des Fensters.
- **Height 0x0006 (6)** — Die Höhe des Fensters.
- **DetailPen 0x0008 (8)** — Die Nummer des Farbregisters, dessen Inhalt als Farbe für die »Details« verwendet wird, also z.B. für die Einzelheiten in den »Gadgets« und der Schrift in der Titelleiste.
- **BlockPen 0x0009 (9)** — Die Nummer des Farbregisters, dessen Inhalt als Farbe für »gröbere« Elemente verwendet wird, also z.B. für das Füllen der Titelleiste.
- **IDCMPFlags 0x000A (10)** — Diese 2-Byte-Marke enthält die *IDCMP-Flags*, mit denen das zugehörige Window initialisiert wird. Eine genaue Beschreibung dieser Flags finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.
- **Flags 0x000E (14)** — Diese 2-Byte-Marke enthält die *Window-Flags*, mit denen das zugehörige Window initialisiert wird. Eine genaue Beschreibung der *Window-Flags* finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.
- **FirstGadget 0x0012 (18)** — Ein Adreßzeiger auf eine lineare Liste von verketteten *Gadget*-Datenstrukturen, die die Gadgets zu diesem Window definieren. Sind keine Gadgets definiert, so steht hier *NULL*. **BEACHTE**: Systemgadgets werden über die *Window-Flags* definiert.
- **CheckMark 0x0016 (22)** — Ein Adreßzeiger auf eine *Image*-Datenstruktur, die das Checkmark zu dem Menü dieses Windows definiert. Steht hier *NULL*, so wird das Default-Checkmark (ein Häkchen) in den Pull-down-Menüs verwendet.
- **Title 0x001A (26)** — Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, die in der Titelleiste des Windows ausgegeben wird. Als Zeichensatz wird der Font des zugehörigen Bildschirms verwendet.
- **Screen 0x001E (30)** — Ist in Types das Bit *CUSTOMSCREEN* gesetzt, so steht hier ein Adreßzeiger auf die zugehörige *Screen*-Datenstruktur.

❑ **BitMap 0x0022 (34)**

Ist in Flags das Refresh-Bit *SUPERBITMAP* gesetzt, so muß hier ein Adreßzeiger auf eine *BitMap*-Datenstruktur stehen, die die Grafikdaten des Windows aufnehmen kann.

❑ **MinWidth 0x0026 (38)**

Hat das Fenster ein Sizing-Gadget, so kann es der Benutzer mit dem Sizing-Gadget bis auf diese minimale Breite verkleinern.

❑ **MinHeight 0x0028 (40)**

Hat das Fenster ein Sizing-Gadget, so kann es der Benutzer mit dem Sizing-Gadget bis auf diese minimale Höhe verkleinern.

❑ **MaxWidth 0x002A (42)**

Hat das Fenster ein Sizing-Gadget, so kann es der Benutzer mit dem Sizing-Gadget bis auf diese maximale Breite verbreitern.

❑ **MaxHeight 0x002C (44)**

Hat das Fenster ein Sizing-Gadget, so kann es der Benutzer mit dem Sizing-Gadget bis auf diese maximale Höhe vergrößern.

❑ **Type 0x002E (46)**

In dieser 2-Byte-Marke stehen die *Screen*-Flags zu diesem Window. Eine genaue Beschreibung dieser Flags finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.

PenPair

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	**Dez**	struct PenPair	PenPair =
		{	RECORD
0000	0	UBYTE DetailPen;	detailPen:UByte;
0001	1	UBYTE BlockPen;	blockPen:UByte;
0002	2	};	END;

Assembler

Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz

»Intuition«: *GListEnv.*

Routinenreferenz

Keine.

Beschreibung

Diese systeminterne Datenstruktur wird in die *GListEnv*-Datenstruktur eingebunden und beinhaltet zwei Nummern von Farbregistern.

❑ **DetailPen 0x0000 (0)**

Die Nummer des Farbregisters, dessen Inhalt als Farbe für die »Details« verwendet wird, also z.B. für die Einzelheiten in den Gadgets und die Schrift in der Titelleiste.

❑ **BlockPen 0x0001 (1)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für »gröbere« Elemente verwendet wird, also z.B. für das Füllen der Titelleiste.

Point

Offset		C (intuition/intuitionbase.h)	Modula-2 (Intuition)
Hex	**Dez**	struct Point	Point =
		{	RECORD
0000	0	SHORT X;	x:INTEGER;
0002	2	SHORT Y;	y:INTEGER;
0004	4	};	END;

Assembler Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz »Intuition«: *IntuitionBase*.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur wird in die *IntuitionBase*-Datenstruktur eingebunden und dient zur Speicherung der Koordinaten eines Punktes.

❑ **X 0x0000 (0)** Die x-Koordinate des Punktes.

❑ **0x0002 (2)** Die y-Koordinate des Punktes.

Preferences

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	**Dez**	struct Preferences	Preferences =
		{	RECORD
0000	0	BYTE FontHeight;	fontHeight:UByte;
0001	1	UBYTE PrinterPort;	printerPort:PrinterPort;
0002	2	USHORT BaudRate;	baudRate:CARDINAL;
0004	4	struct timeval KeyRptSpeed;	keyRptSpeed:TimeVal;
000C	12	struct timeval KeyRptDelay;	keyRptDelay:TimeVal;
0014	20	struct timeval DoubleClick;	doubleClick:TimeVal;
001C	28	USHORT PointerMatrix[36];	pointerMatrix:ARRAY[0..35] OF CARDINAL;
0064	100	BYTE XOffset;	xOffset:Byte;
0065	101	BYTE YOffset;	yOffset:Byte;

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
0066	102	USHORT color17;	color17:CARDINAL;
0068	104	USHORT color18;	color18:CARDINAL;
006A	106	USHORT color19;	color19:CARDINAL;
006C	108	USHORT PointerTicks;	pointerTicks:CARDINAL;
006E	110	USHORT color0;	color0:CARDINAL;
0070	112	USHORT color1;	color1:CARDINAL;
0072	114	USHORT color2;	color2:CARDINAL;
0074	116	USHORT color3;	color3:CARDINAL;
0076	118	BYTE ViewXOffset;	viewXOffset:Byte;
0077	119	BYTE ViewYOffset;	viewYOffset:Byte;
0078	120	WORD ViewInitX;	viewInitX:INTEGER;
007A	122	WORD ViewInitY;	viewInitY:INTEGER;
007C	124	BOOL EnableCLI;	enableCLI:CARDINALI;
007E	126	USHORT PrinterType;	printerType:CARDINALI;
0080	128	UBYTE PrinterFilename[30];	printerFilename:ARRAY [0..29] OF CHAR;
009E	158	USHORT PrintPitch;	printPitch:CARDINAL;
00A0	160	USHORT PrintQuality;	printQuality:CARDINAL;
00A2	162	USHORT PrintSpacing;	printSpacing:CARDINAL;
00A4	164	UWORD PrintLeftMargin;	printLeftMargin:CARDINAL;
00A6	166	UWORD PrintRightMargin;	printRightMargin:CARDINAL;
00A8	168	USHORT PrintImage;	printImage:CARDINAL;
00AA	170	USHORT PrintAspect;	printAspect:CARDINAL;
00AC	172	USHORT PrintShade;	printShade:CARDINAL;
00AE	174	WORD PrintThreshold;	printThreshold:INTEGER;
00B0	176	USHORT PaperSize;	paperSize:CARDINAL;
00B2	178	UWORD PaperLength;	paperLength:CARDINAL;
00B4	180	USHORT PaperType;	paperType:CARDINAL;
00B6	182	UBYTE SerRWBits;	serRWBits:UByte;
00B7	183	UBYTE SerStopBuf;	serStopBuf:UByte;
00B8	184	UBYTE SerParShk;	serParShk:SerParShkSet;
00B9	185	UBYTE LaceWB;	laceWB:BOOLEAN;
00BA	186	UBYTE WorkName[30];	workName:ARRAY[0..29] OF CHAR;
00D8	216	BYTE RowSizeChange;	padding:ARRAY [0..15] OF BYTE;
00D9	217	BYTE ColumnSizeChange;	
00DA	218	UWORD PrintFlags;	
00DC	220	UWORD PrintMaxWidth;	
00DE	222	UWORD PrintMaxHeigth;	
00E0	224	UBYTE PrintDensity;	
00E1	225	UBYTE PrintXOffset;	
00E2	226	UWORD wb_Width;	
00E4	228	UWORD wb_Heigth;	

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
00E6	230	UBYTE wb_Depth;	
00E7	231	UBYTE ext_Size;	
00E8	232	};	END;

Assembler Präfix: *pf_*

Datenstrukturenreferenz »Intuition«: *IntuitionBase.*

»PrtBase«: *PrinterData.*

Routinenreferenz »Intuition«: *GetDefPrefs, GetPrefs, SetPrefs.*

4

Beschreibung

In der *Preferences*-Datenstruktur werden die von dem Programm »Preferences« bekannten Voreinstellungen abgelegt.

❑ **FontHeight 0x0000 (0)** Folgende zwei Konstanten geben die Schriftbreite des Systemzeichensatzes an:

C	Modula-2	Hex	Beschreibung
TOPAZ_EIGHTY	topazEighty	0008	Einstellen des Systemzeichensatzes auf 80 Zeichen pro Zeile.
TOPAZ_SIXTY	topazSixty	0009	Einstellen des Systemzeichensatzes auf 60 Zeichen pro Zeile.

❑ **PrinterPort 0x0001 (1)** Folgende Bits sind für dieses Feld definiert:

C	Modula-2	Hex	Beschreibung
PARALLEL_PRINTER	parallelPrinter	0000	Drucker befindet sich am Parallel-Port.
SERIAL_PRINTER	serialPrinter	0001	Drucker befindet sich am Serial-Port.

❑ **BaudRate 0x0002 (2)** Folgende Konstanten definieren die verschiedenen Baudraten (Baud = übertragende Bits pro Sekunde) der seriellen Schnittstelle:

C	Modula-2	Hex	Beschreibung
BAUD_110	baud110	0000	Setzt die Baudrate auf 110 Bits/s.
BAUD_300	baud300	0001	Setzt die Baudrate auf 300 Bits/s.
BAUD_1200	baud1200	0002	Setzt die Baudrate auf 1200 Bits/s.
BAUD_2400	baud2400	0003	Setzt die Baudrate auf 2400 Bits/s.
BAUD_4800	baud4800	0004	Setzt die Baudrate auf 4800 Bits/s.
BAUD_9600	baud9600	0005	Setzt die Baudrate auf 9600 Bits/s.
BAUD_19200	baud19200	0006	Setzt die Baudrate auf 19200 Bits/s.
BAUD_MIDI	baudMidi	0007	Setzt die Baudrate auf die Midi-Norm.

❑ **KeyRptSpeed 0x0004 (4)**
Diese eingebundene *TimeVal*-Datenstruktur definiert die Zeitspanne, die bei der Tastenwiederholung (Gedrückthalten einer Taste) zwischen der Ausgabe der einzelnen Zeichen liegt.

❑ **KeyRptDelay 0x000C (12)**
Diese eingebundene *TimeVal*-Datenstruktur definiert die Zeitspanne, die das System beim Drücken einer Taste wartet, bevor sie ständig wiederholt wird.

❑ **DoubleClick 0x0014 (20)**
Diese eingebundene *TimeVal*-Datenstruktur definiert die Zeitspanne, die zwischen zwei »Maus-Klicks« liegen darf, damit dies als »Doppel-Klick« erkannt wird.

❑ **PointerMatrix 0x001C (28)**
Hier sind die Grafikdaten eingebunden, die den Standard-Mauszeiger als Sprite definieren.

❑ **XOffset 0x0064 (100)**
Der x-Offset des Hot-Spots in dem Mauszeiger.

❑ **YOffset 0x0065 (101)**
Der y-Offset des Hot-Spots in dem Mauszeiger.

❑ **color17 0x0066 (102)**
Der Farbwert, der als erste Farbe für die Darstellung des Mauszeigers verwendet wird.

❑ **color18 0x0068 (104)**
Der Farbwert, der als zweite Farbe für die Darstellung des Mauszeigers verwendet wird.

❑ **color19 0x006A (106)**
Der Farbwert, der als dritte Farbe für die Darstellung des Mauszeigers verwendet wird.

❑ **PointerTicks 0x006C (108)**
Dieser Wert beschreibt den Übersetzungsfaktor der Mausbewegungen. Die sinnvollen Werte sind die Zweierpotenzen 1, 2, oder 4. Für größere Werte fordert die Maus viel Bewegungsfreiheit, für 64 bereits eine Fläche von der Größe einer Tischtennisplatte.

❑ **color0 0x006E (110)**
Der Farbwert, der als Default-Hintergrundfarbe verwendet wird.

❑ **color1 0x0070 (112)**
Der Farbwert, der als Default-Vordergrundfarbe (Farbe der Windows und des Textes) verwendet wird.

- **color2 0x0072 (114)** Der Farbwert, der als dritte Default-Farbe (z.B. in den Close- und Depth-Gadgets) verwendet wird.
- **color3 0x0074 (116)** Der Farbwert, der als vierte Default-Farbe (z.B. für den Standard-Textcursor) verwendet wird.
- **ViewXOffset 0x0076 (118)** Der x-Offset des Gesamtdisplays.
- **ViewYOffset 0x0077 (119)** Der y-Offset des Gesamtdisplays.
- **ViewInitX 0x0078 (120)** Der x-Initialisierungswert für alle Views.
- **ViewInitY 0x007A (122)** Der y-Initialisierungswert für alle Views.
- **EnableCLI 0x007C (124)** Ein Boolescher Ausdruck, der angibt, ob das »CLI-Icon« auf dem Display erscheinen soll (TRUE) oder nicht (FALSE). In den neueren Versionen wird dieser Wert jedoch nicht mehr beachtet, das Icon erscheint immer.
- **PrinterType 0x007E (126)** Folgende Konstanten definieren die verschiedenen Druckertypen, die von den Preferences unterstützt werden:

C	Modula-2	Hex	Beschreibung
CUSTOM_NAME	customName	0000	Ein Drucker, der nicht mit einem der unten aufgeführten kompatibel ist.
ALPHA_P_101	alphaP101	0001	Ein Alphacom oder Alphaprp-101-Typenraddrucker.
BROTHER_15XL	brother15XL	0002	Ein Brother-HR-15XL-Typenraddrucker.
CBM_MPS1000	cbmMps1000	0003	Ein Commodore-MPS-1000-Matrixdrucker.
DIAB_630	diab630	0004	Ein Diablo-630-Typenraddrucker.
DIAB_ADV_D25	diabAdvD25	0005	Ein Diablo Advantage-D251-Typenraddrucker.
DIAB_C_150	diabC150	0006	Ein Diablo-C-150-Farbdrucker.
EPSON	epson	0007	Ein Epson-Drucker der RX/FX-Reihe.
EPSON_JX_80	epsonJX80	0008	Ein Epson-JX-80-Farbdrucker
OKIMATE_20	okimate20	0009	Ein Okimate-20-Farbdrucker.
QUME_LP_20	QumeLP20	000A	Ein Qume-LetterPro-20-Typenraddrucker.
HP_LASERJET	hpLaserjet	000B	Ein Hewlett-Packard-LaserJet-Laserdrucker.
HP_LASERJET _PLUS	hpLaserjet Plus	000C	Ein Hewlett-Packard-LaserJet-Plus-Laserdrucker.

- **PrinterFilename 0x0080 (128)** Der Name des Druckertreibers, falls in PrinterType *CUSTOM* eingestellt ist. Der Druckertreiber muß sich in dem Unterverzeichnis »devs/printers« befinden.

❑ **PrintPitch 0x009E (158)** Die momentan eingestellte Zeichendichte. Dieses Feld kann die Werte 10, 12 oder 15 annehmen. Das entspricht den Schriftarten Pica, Elite oder Fine und auf DIN A4 80, 96 oder 120 Zeichen pro Zeile.

❑ **PrintQuality 0x00A0 (160)** Die folgenden Konstanten definieren die Druckbildqualität.

C	Modula-2	Hex	Beschreibung
DRAFT	draft	0000	Draft-Schrift (Ausdruck mit der Standard-Schriftart)
LETTER	letter	0100	NLQ-Schrift (Ausdruck mit der »Near Letter Quality«-Schriftart)

❑ **PrintSpacing 0x00A2 (162)** Die folgenden Konstanten definieren den Zeilenabstand.

C	Modula-2	Hex	Beschreibung
SIX_LPI	sixLPI	0000	Setzt den Zeilenabstand auf 6 LPI (Lines per Inch).
EIGHT_LPI	eightLPI	0200	Setzt den Zeilenabstand auf 8 LPI (Lines per Inch).

❑ **PrintLeftMargin 0x00A4 (164)** Der momentan eingestellte linke Rand, also ab der wievielten Spalte bei der Ausgabe auf den Drucker der Text gedruckt wird.

❑ **PrintRightMarging 0x00A6 (166)** Der momentan eingestellte rechte Rand, also bis zur wievielten Spalte bei der Ausgabe auf den Drucker der Text gedruckt wird.

❑ **PrintImage 0x00A8 (168)** Die folgenden Werte definieren bei einem Grafikausdruck das Druckbild.

C	Modula-2	Hex	Beschreibung
IMAGE_POSITIVE	imagePositive	0000	Nur die gelöschten Bits werden ausgedruckt.
IMAGE_NEGATIVE	imageNegative	0001	Nur die gesetzten Bits werden ausgedruckt.

❑ **PrintAspect 0x00AA (170)** Die folgenden Werte definieren bei einem Grafikausdruck die Lage des Druckbildes.

C	Modula-2	Hex	Beschreibung
ASPECT_HORIZ	aspectHoriz	0000	Die Grafik wird horizontal (normal) ausgedruckt.
ASPECT_VERT	aspectVert	0001	Die Grafik wird vertikal (um 90 Grad gedreht) ausgedruckt.

❑ **PrintShade 0x00AC (172)** Die folgenden Werte definieren bei einem Grafikausdruck die Schattierung.

C	Modula-2	Hex	Beschreibung
SHADE_BW	shadeBW	0000	In Abhängigkeit von PrintImage werden gelöschte, bzw. gesetzte Bits, schwarz auf weiß ausgedruckt.
SHADE_GREYSCALE	shade Greyscale	0001	Erstreckt sich die Grafik über mehrere Bitplanes, so werden die Farben in verschiedene Grautöne umgerechnet.
SHADE_COLOR	shadeColor	0002	Ist ein Farbdrucker angeschlossen, so wird die Grafik in Farbe ausgedruckt.

❑ **PrintThreshold 0x00AE (174)** Dieser Wert bestimmt im *PrintShade*-Modus SHADE_GREYSCALE die allgemeine Graustufe. Sinnvolle Werte liegen zwischen 1 und 15.

❑ **PaperSize 0x00B0 (176)** Die folgenden Werte definieren das Format des verwendeten Druckerpapiers.

C	Modula-2	Hex	Beschreibung
US_LETTER	usLetter	0000	Amerikanische Brief-Norm.
US_LEGAL	usLegal	0010	Amerikanische Amts-Norm.
N_TRACTOR	nTractor	0020	Papierbreite maximal ca. 25 cm.
W_TRACTOR	wTractor	0030	Papierbreite maximal ca. 35 cm.
CUSTOM	custom	0040	Keine der oben genannten.

❑ **PaperLength 0x00B2 (178)** Ist in *PaperSize* das Flag *CUSTOM* gesetzt, so muß hier die Seitenlänge des Druckerpapiers in Zeilen stehen.

❑ **PaperType 0x00B4 (180)** Die folgenden Werte definieren die Art des Druckerpapiers.

4

C	Modula-2	Hex	Beschreibung
FANFOLD	fanfold	0000	Es wird Endlospapier verwendet.
SINGLE	single	0080	Es wird Einzelblattpapier verwendet.

❑ **SerRWBits 0x00B6 (182)**

Die folgenden Werte definieren in dieser 1-Byte-Marke für die serielle Schnittstelle die Anzahl der Read- und Write-Bits.

C	Modula-2	Hex	Beschreibung
SWRITE_BITS	writeBits	000F	Das untere Nibbel enthält die Anzahl der zu schreibenden Bits.
SREAD_BITS	readBits	00F0	Das obere Nibbel enthält die Anzahl der zu lesenden Bits.

❑ **SerStopBuf 0x00B7 (183)**

Die folgenden Werte definieren in dieser 1-Byte-Marke für die serielle Schnittstelle die Anzahl der Stop-Bits.

C	Modula-2	Hex	Beschreibung
SBUFSIZE_BITS	bufSizeBits	000F	Das untere Nibbel enthält die Größe des Datenpuffers.
SSTOP_BITS	stopBits	00F0	Das obere Nibbel enthält die Anzahl der Stop-Bits.

❑ **SerParShk 0x00B8 (184)**

Das untere Nibbel dieses Werts enthält den Handshake-Modus, das obere die Parität. Folgende Flags können gesetzt werden:

C	Modula-2	Hex	Beschreibung
SHSHAKE_XON	shakeXon	0000	Handshake-Modus xOn/xOff.
SHSHAKE_RTS	shakeRts	0001	Handshake-Modus RTS/CTS.
SHSHAKE_NONE	shakeNone	0002	Kein Handshake Modus.
SPARITY_BITS		00F0	Das obere Nibbel für die Parität.
SPARITY_NONE	parityNone	0000	Keine Parität.
SPARITY_EVEN	parityEven	0001	Gerade Parität.
SPARITY_ODD	parityOdd	0002	Ungerade Parität.

- **LaceWB 0x00B9 (185)** Wird die Workbench im Interlace-Modus dargestellt, so steht hier 1, sonst 0.
- **WorkName 0x00BA (186)** Ein 30-Byte langer Zwischenspeicher für den Druckernamen.
- **RowSizeChange 0x00D8 (216)** Größenveränderungsfaktor für die Zeilen.
- **ColumnSizeChange 0x00D9 (217)** Größenveränderungsfaktor für die Spalten.
- **PrintFlags 0x00DA (218)** In dieser 2-Byte-Marke können durch den Benutzer eigene Druckerflags eingeführt werden.
- **PrintMaxWidth 0x00DC (220)** Die maximale Breite eines Grafikausdrucks in zehntel Inch.
- **PrintMaxHeigth 0x00DE (222)** Die maximale Höhe eines Grafikausdrucks in zehntel Inch.
- **PrintDensity 0x00E0 (224)** Die Zeichendichte, mit der die Grafik ausgedruckt werden soll.
- **PrintXOffset 0x00E1 (225)** Der linke Rand bei einem Grafikausdruck in zehntel Inch.
- **wb_Width 0x00E2 (226)** Die Breite des Workbench-Screens (nicht veränderbar).
- **wb_Heigth 0x00E4 (228)** Die Höhe des Workbench-Screens.
- **wb_Depth 0x00E6 (230)** Die Tiefe (Anzahl der Bitplanes) des Workbench-Screens.
- **Size 0x00E7 (231)** Interner Systemparameter, der nicht verändert werden darf.

PropInfo

Offset		**C (intuition/intuition.h)**	**Modula-2 (Intuition)**
Hex	**Dez**	struct PropInfo	PropInfo =
		{	RECORD
0000	0	USHORT Flags;	flags:PropInfoFlagSet;
0002	2	USHORT HorizPot;	horizPot:CARDINAL;
0004	4	USHORT VertPot;	vertPot:CARDINAL;
0006	6	USHORT HorizBody;	horizBody:CARDINAL;
0008	8	USHORT VertBody;	vertBody:CARDINAL;
000A	10	USHORT CWidth;	cWidth:CARDINAL;
000C	12	USHORT CHeight;	cHeight:CARDINAL;
000E	14	USHORT HPotRes;	hPotRes:CARDINAL;
0010	16	USHORT VPotRes;	vPotRes:CARDINAL;
0012	18	USHORT LeftBorder;	leftBorder:CARDINAL;
0014	20	USHORT TopBorder;	topBorder:CARDINAL;
0016	22	};	END;

Assembler Präfix: *pi_*

Datenstrukturenreferenz »Intuition«: *Gadget, IntuitionBase.*

Routinenreferenz Keine.

Beschreibung

Dies ist eine zusätzliche Datenstruktur für Proportional-Gadgets. Bei dieser Art von Gadgets kann ein zugehöriger Wert über einen verstellbaren »Knopf«, ähnlich wie bei einem Schieberegler (oder -Potentiometer), eingestellt werden. Der Bereich, in dem der Knopf bewegt werden kann, ist die innere Fläche des Gadgets und heißt »Container«. Typische Anwendungen der Proportional-Gadgets sind Farbauswahl oder Positionseinstellungen in langen Texten.

❑ **Flags 0x0000 (0)** Folgende Bits dieser 2-Byte-Marke sind definiert:

C	Modula-2	Hex	Beschreibung
AUTOKNOB	autoKnob	0001	Der Knopf ist ein Rechteck, dessen Größe ebenfalls einen Parameter proportional darstellt.
FREEHORIZ	freeHoriz	0002	Der Knopf kann horiz. bewegt werden.
FREEVERT	freeVert	0004	Der Knopf kann vertikal bewegt werden.
PROPBORDERLESS	propBorderless	0008	Es wird kein Rahmen um den Bereich gezeichnet, in dem der Knopf bewegt werden kann.
KNOBHIT	knobHit	0100	Der Knopf wird gerade bewegt.

❑ **HorizPot 0x0002 (2)** Momentan eingestellter Wert in horizontaler Richtung. Dabei ist der kleinste Wert Null und der größte 0xFFFF (65535), es müssen also gegebenenfalls Anpassungen vorgenommen werden, um Werte in den gewünschten Bereichen (z.B. zwischen 0 und 15 bei Farbeinstellungen) zu erhalten. Inwiefern dieser Wert veränderbar ist, wird durch *FREEHORIZ* in *Flags* definiert.

❑ **VertPot 0x0004 (4)** Momentan eingestellter Wert in vertikaler Richtung. Dabei ist der kleinste Wert Null und der größte 0xFFFF (65535), es müssen also gegebenenfalls Anpassungen vorgenommen werden, um Werte in den gewünschten Bereichen zu erhalten. Inwiefern dieser Wert veränderbar ist, wird durch *FREEVERT* in *Flags* definiert.

❑ **HorizBody 0x0006 (6)** Der hier eingestellte Wert bestimmt die Schrittweite von *HorizPot*. Ist das Flag *AUTOKNOB* in *Flags* gesetzt, so bestimmt dieser Wert auch die Breite des Knopfes.

❑ **VertBody 0x0008 (8)** — Der hier eingestellte Wert bestimmt die Schrittweite von *VertPot*. Ist das Flag *AUTOKNOB* in *Flags* gesetzt, so bestimmt dieser Wert auch die Höhe des Knopfes.

❑ **CWidth 0x000A (10)** — Hier trägt »Intuition« die aktuelle Breite des Containers ein.

❑ **CHeight 0x000C (12)** — Hier trägt »Intuition« die aktuelle Höhe des Containers ein.

❑ **HPotRes 0x000E (14)** — Hier trägt »Intuition« die Anzahl der in horizontaler Richtung zu erhaltenden Werte ein.

❑ **VPotRes 0x0010 (16)** — Hier trägt »Intuition« die Anzahl der in vertikaler Richtung zu erhaltenden Werte ein.

❑ **LeftBorder 0x0012 (18)** — Abstand des Containers vom linken Rand des zugehörigen Proportional-Gadgets.

❑ **TopBorder 0x0014 (20)** — Abstand des Containers vom oberen Rand des zugehörigen Proportional-Gadgets.

Remember

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	**Dez**	struct Remember {	Remember = RECORD
0000	0	struct Remember *NextRemember;	nextRemember:RememberPtr;
0004	4	ULONG RememberSize;	rememberSize:LONGCARD;
0008	8	UBYTE *Memory;	memory:ADDRESS;
000C	12	};	END;

Assembler — Präfix: *rm_*.

Datenstrukturenreferenz — »Intuition«: *IntuitionBase*, *Remember*.

Routinenreferenz — »Intuition«: *AllocRemember*, *FreeRemember*.

Beschreibung

Mit Hilfe der *Remember*-Datenstruktur werden Speicherbereiche für die verschiedensten Anwendungen reserviert. Dabei wird eine Liste von mehreren miteinander verketteten *Remember*-Datenstrukturen gebildet, die mit einem einzigen Aufruf der Routine *FreeRemember* wieder freigegeben werden kann.

❑ **NextRemember 0x0000 (0)** — Über diesen Adreßzeiger können mehrere *Remember*-Datenstrukturen zu einer linearen Liste verknüpft werden. Folgt keine weitere *Remember*-Datenstruktur, so steht hier *NULL*.

❑ **RememberSize 0x0004 (4)** — Die Größe des Speicherbereichs.

❑ **Memory 0x0008 (8)** — Der Adreßzeiger auf den Speicherbereich.

4

Requester

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct Requester {	Requester = RECORD
0000	0	struct Requester *OlderRequest;	olderRequest:RequesterPtr;
0004	4	SHORT LeftEdge;	leftEdge:INTEGER;
0006	6	SHORT TopEdge;	topEdge:INTEGER;
0008	8	SHORT Width;	width:INTEGER;
000A	10	SHORT Height;	height:INTEGER;
000C	12	SHORT RelLeft;	relLeft:INTEGER;
000E	14	SHORT RelTop;	relTop:INTEGER;
0010	16	struct Gadget *ReqGadget;	reqGadget:GadgetPtr;
0014	20	struct Border *ReqBorder;	reqBorder:BorderPtr;
0018	24	struct IntuiText *ReqText;	reqText:IntuiTextPtr;
001C	28	USHORT Flags;	flags:RequesterFlagSet;
001E	30	UBYTE BackFill;	backFill:UByte;
0020	32	struct Layer *ReqLayer;	reqLayer:LayerPtr;
0024	36	UBYTE ReqPad1[32];	reqPad1:ARRAY [0..31] OF BYTE;
0044	68	struct BitMap *ImageBMap;	imageBMap:BitMapPtr;
0048	72	struct Window *RWindow;	rWindow:WindowPtr;
004C	76	UBYTE ReqPad2[36];	reqPad2:ARRAY [0..35] OF BYTE;
0070	112	};	END;

Assembler Präfix: *rq_*

Datenstrukturenreferenz »Intuition«: *GListEnv, Requester, Window.*

Routinenreferenz »Intuition«: *ActivateGadget, AddGList, EndRequest, InitRequester, ModifyProp, NewModifyProp, OffGadget, OnGadget, RefreshGadgets, RefreshGList, Request, SetDMRequest.*

Beschreibung

Wird ein Requester geöffnet, so stehen in der zugehörigen *Requester*-Datenstruktur die entsprechenden Informationen über die Darstellung des Requesters.

- **OlderRequest 0x0000 (0)** — Ist bereits ein Requester geöffnet, so trägt »Intuition« hier einen Adreßzeiger auf den zuletzt geöffneten ein.

- **LeftEdge 0x0004 (4)** — Ist das *POINTREL*-Bit in *Flags* gelöscht, so definiert dieser Wert den Abstand des Requesters vom linken Rand des zugehörigen Windows.

- **TopEdge 0x0006 (6)** Ist das *POINTREL*-Bit in *Flags* gelöscht, so definiert dieser Wert den Abstand des Requesters vom oberen Rand des zugehörigen Windows.

- **Width 0x0008 (8)** Die Breite des gesamten Requesters, also einschließlich der dazugehörigen Texte und Gadgets.

- **Height 0x000A (10)** Die Höhe des gesamten Requesters, also einschließlich der dazugehörigen Texte und Gadgets.

- **RelLeft 0x000C (12)** Ist das *POINTREL*-Bit in *Flags* gesetzt, so ist dies der horizontale Abstand des Requesters, relativ zur Position des Mauszeigers zum Zeitpunkt der Auslösung.

- **RelTop 0x000E (14)** Ist das *POINTREL*-Bit in *Flags* gesetzt, so ist dies der vertikale Abstand des Requesters, relativ zur Position des Mauszeigers zum Zeitpunkt der Auslösung.

- **ReqGadget 0x0010 (16)** Ein Adreßzeiger auf die lineare Liste von *Gadget*-Datenstrukturen, die zu diesem Requester definiert sind. **ACHTUNG**: Hier darf nicht *NULL* stehen. In der letzten *Gadget*-Datenstruktur **muß** in *Flags* das *ENDGADGET*-Bit gesetzt sein, damit sich das Requester wieder schließen kann.

- **ReqBorder 0x0014 (20)** Ein optionaler Adreßzeiger auf eine *Border*-Datenstruktur, die dieses Requester umrandet.

- **ReqText 0x0018 (24)** Ein optionaler Adreßzeiger auf eine *IntuiText*-Datenstruktur, die den in diesem Requester auszugebenden Text definiert.

- **Flags 0x001C (28)** Folgende Flags sind für diese 2-Byte-Marke definiert:

C	Modula-2	Hex	Beschreibung
POINTREL	pointRel	0001	Die Position des Requesters ist relativ zur Position des Mauszeigers.
PREDRAWN	preDrawn	0002	Zu diesem Requester existiert eine eigene Bitmap.
NOISYREQ	noisyReq	0004	Durch Setzen dieses Bits wird ermöglicht, daß weiterhin Tastatur- und Maus-Events auftreten können.
REQOFFWINDOW	reqOffWindow	1000	Dieses Bit wird von »Intuition« gesetzt, wenn sich mindestens ein Requester außerhalb des zugehörigen Fensters befindet.
REQACTIVE	reqActive	2000	Das Requester ist aktiv.
SYSREQUEST	sysRequest	4000	Ein System-Requester ist aktiv.
DEFERREFRESH	deferRefresh	8000	Ein Refresh wurde unterbrochen.

- **BackFill 0x001E (30)** Mit der hier stehenden Pen-Nummer können Sie die Hintergrundfarbe des Requesters wählen.
- **ReqLayer 0x0020 (32)** Ein Adreßzeiger auf die *Layer*-Datenstruktur, die bei der Ausgabe dieses Requesters verwendet wird.
- **ReqPad1 0x0024 (36)** Von »Intuition« intern verwendeter Zwischenspeicher.
- **ImageBMap 0x0044 (68)** Ist das Bit *PREDRAWN* in Flags gesetzt, so muß hier ein Adreßzeiger auf eine eigene *BitMap*-Datenstruktur stehen.
- **RWindow 0x0048 (72)** Von »Intuition« intern verwendeter Systemparameter.
- **ReqPad2 0x004C (76)** Von »Intuition« intern verwendeter Zwischenspeicher.

Screen

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	**Dez**	struct Screen	Screen =
		{	RECORD
0000	0	struct Screen *NextScreen;	nextScreen:ScreenPtr;
0004	4	struct Window *FirstWindow;	firstWindow:WindowPtr;
0008	8	SHORT LeftEdge;	leftEdge:INTEGER;
000A	10	SHORT TopEdge;	topEdge:INTEGER;
000C	12	SHORT Width;	width:INTEGER;
000E	14	SHORT Height;	height:INTEGER;
0010	16	SHORT MouseY;	mouseY:INTEGER;
0012	18	SHORT MouseX;	mouseX:INTEGER;
0014	20	USHORT Flags;	flags:ScreenFlagSet;
0016	22	UBYTE *Title;	title:ADDRESS;
001A	26	UBYTE *DefaultTitle;	defaultTitle:ADDRESS;
001E	30	BYTE BarHeight;	barHeight:Byte;
001F	31	BYTE BarVBorder;	barVBorder:Byte;
0020	32	BYTE BarHBorder;	barHBorder:Byte;
0021	33	BYTE MenuVBorder;	menuVBorder:Byte;
0022	34	BYTE MenuHBorder;	menuHBorder:Byte;
0023	35	BYTE WBorTop;	wBorTop:Byte;
0024	36	BYTE WBorLeft;	wBorLeft:Byte;
0025	37	BYTE WBorRight;	wBorRight:Byte;
0026	38	BYTE WBorBottom;	wBorBottom:Byte;
0028	40	struct TextAttr *Font;	font:TextAttrPtr;
002C	44	struct ViewPort ViewPort;	viewPort:ViewPort;
0054	84	struct RastPort RastPort;	rastPort:RastPort;
00B8	184	struct BitMap BitMap;	bitMap:BitMap;
00E0	224	struct Layer_Info LayerInfo;	layerInfo:LayerInfo;

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
0146	326	struct Gadget *FirstGadget;	firstGadget:GadgetPtr;
014A	330	UBYTE DetailPen;	detailPen:UByte;
014B	331	UBYTE BlockPen;	blockPen:UByte;
014C	332	USHORT SaveColor0;	saveColor0:CARDINAL;
014E	334	struct Layer *BarLayer;	barLayer:LayerPtr;
0152	338	UBYTE *ExtData;	extData:ADDRESS;
0156	342	UBYTE *UserData;	userData:ADDRESS;
015A	346	};	END;

4

Assembler Präfix: *sc_*

Datenstrukturenreferenz »Intuition«: *GListEnv*, *IntuitionBase*, *NewWindow*, *Screen*, *Window*.

Routinenreferenz »Intuition«: *CloseScreen*, *DisplayBeep*, *GetScreenData*, *MakeScreen*, *ModifyProp*, *MoveScreen*, *OpenScreen*, *ScreenToBack*, *ScreenToFront*, *Show Title*.

Beschreibung

Wird durch die *OpenScreen*-Routine ein Screen geöffnet, so gibt sie einen Zeiger auf die zugehörige *Screen*-Datenstruktur zurück. In dieser finden sie die wichtigsten Informationen über den Screen.

- **NextScreen 0x0000 (0)** Sind mehrere Screens geöffnet, so werden die dazugehörigen *Screen*-Datenstrukturen über diesen Adreßzeiger linear miteinander verknüpft. Dabei gibt die Position des Screens in dieser Liste seine Tiefe an. Der Screen, der momentan hinter allen anderen liegt, steht also am Ende der Liste und hat an dieser Stelle den Wert *NULL*. Einen Adreßzeiger auf den vorderen, ersten Screen der Liste steht in der *IntuitionBase*-Datenstruktur.
- **FirstWindow 0x0004 (4)** Ist dieser Wert ungleich *NULL*, so ist dies ein Adreßzeiger auf die *Window*-Datenstruktur auf das erste Fenster, das auf diesem Screen geöffnet ist.
- **LeftEdge 0x0008 (8)** Die x-Position der linken oberen Ecke des Screens muß in den vorliegenden Versionen immer auf Null gesetzt werden.
- **TopEdge 0x000A (10)** Die y-Position der linken oberen Ecke des Screens.
- **Width 0x000C (12)** Die Breite des Screens muß, je nach Auflösung, 320 oder 640 sein.
- **Height 0x000E (14)** Die Höhe des Screens.

4

- **MouseY 0x0010 (16)** — Die augenblickliche y-Koordinate der Position des Mauszeigers auf diesem Screen, relativ zu seiner linken oberen Ecke.
- **MouseX 0x0012 (18)** — Die augenblickliche x-Koordinate der Position des Mauszeigers auf diesem Screen, relativ zu seiner linken oberen Ecke.
- **Flags 0x0014 (20)** — In dieser 2-Byte-Marke stehen die Screen-Flags. Eine genaue Beschreibung dieser Flags finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.
- **Title 0x0016 (22)** — Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, deren Inhalt auf der Titelleiste ausgegeben wird.
- **DefaultTitle 0x001A (26)** — Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, deren Inhalt als Default-Titel erhalten bleibt.
- **BarHeight 0x001E (30)** — Die Höhe der Titelleiste.
- **BarVBorder 0x001F (31)** — Die Breite des vertikalen Bildschirmrandes.
- **BarHBorder 0x0020 (32)** — Die Höhe des horizontalen Bildschirmrandes.
- **MenuVBorder 0x0021 (33)** — Die Breite der vertikalen Ränder der zu diesem Screen definierten Pull-down-Menüs. Dieser Wert gilt auch für die Pull-down-Menüs, die zu den Windows auf diesem Screen gehören.
- **MenuHBorder 0x0022 (34)** — Die Höhe der horizontalen Ränder der zu diesem Screen definierten Pull-down-Menüs. Dieser Wert gilt auch für die Pull-down-Menüs, die zu den Windows auf diesem Screen gehören.
- **WBorTop 0x0023 (35)** — Dieser Wert definiert für alle Fenster, die auf diesem Screen geöffnet werden, die Höhe des oberen Fensterrandes.
- **WBorLeft 0x0024 (36)** — Dieser Wert definiert für alle Fenster, die auf diesem Screen geöffnet werden, die Breite des linken Fensterrandes.
- **WBorRight 0x0025 (36)** — Dieser Wert definiert für alle Fenster, die auf diesem Screen geöffnet werden, die Breite des rechten Fensterrandes.
- **WBorBottom 0x0026 (37)** — Dieser Wert definiert für alle Fenster, die auf diesem Screen geöffnet werden, die Höhe des unteren Fensterrandes.
- **Font 0x0028 (40)** — Ein Adreßzeiger auf die *TextAttr*-Datenstruktur des Default-Fonts, den für diesen Screen voreingestellten Zeichensatz. Dieser steht dann auch den Displayelementen (z.B. Window) als solcher zur Verfügung.
- **ViewPort 0x002C (44)** — Einbindung einer *ViewPort*-Datenstruktur, die bei der Darstellung dieses Screens benutzt wird.
- **RastPort 0x0054 (84)** — Einbindung einer *RastPort*-Datenstruktur, die den »Zeichenstift« definiert, der für diesen Screen bereitgehalten wird.

- **BitMap 0x00B8 (184)** Einbindung einer *BitMap*-Datenstruktur zur Bestimmung des Displayspeichers für diesen Screen.
- **LayerInfo 0x00E0 (224)** Einbindung einer *LayerInfo*-Datenstruktur.
- **FirstGadget 0x0146 (326)** Dieser Parameter wird von den vorliegenden Versionen noch nicht unterstützt.
- **DetailPen 0x014A (330)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für die »Details« verwendet wird, also z.B. für die Einzelheiten in den Gadgets und die Schrift in der Titelleiste.
- **BlockPen 0x014B (331)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für »gröbere« Elemente verwendet wird, also z.B. die Titelleiste.
- **SaveColor0 0x014C (332)** Ein Zwischenspeicher für die Nummer eines Farbregisters. Dieser wird beim Blinken des Screens verwendet.
- **BarLayer 0x014E (334)** Ein Adreßzeiger auf eine *Layer*-Datenstruktur, die zur Darstellung der Menüleisten gebraucht wird.
- **ExtData 0x0152 (338)** Ein unbenutzter Adreßzeiger für spätere Erweiterungen.
- **UserData 0x0156 (342)** Dieser Adreßzeiger steht dem Benutzer zur Verfügung. Über ihn können jedem Screen beliebige Zusatzdaten zugeordnet werden.

StringInfo

Offset Hex	Offset Dez	C (intuition/intuition.h)	Modula-2 (Intuition)
		struct StringInfo	StringInfo =
		{	RECORD
0000	0	UBYTE *Buffer;	buffer:ADDRESS;
0004	4	UBYTE *UndoBuffer;	undoBuffer:ADDRESS;
0008	8	SHORT BufferPos;	bufferPos:INTEGER;
000A	10	SHORT MaxChars;	maxChars:INTEGER;
000C	12	SHORT DispPos;	dispPos:INTEGER;
000E	14	SHORT UndoPos;	undoPos:INTEGER;
0010	16	SHORT NumChars;	numChars:INTEGER;
0012	18	SHORT DispCount;	dispCount:INTEGER;
0014	20	SHORT CLeft;	cLeft:INTEGER;
0016	22	SHORT CTop;	cTop:INTEGER;
0018	24	struct Layer *LayerPtr;	layerPtr:LayerPtr;
001C	28	LONG LongInt;	longInt:LONGINT;
0020	32	struct KeyMap *AltKeyMap;	altKeyMap:KeyMapPtr;
0024	36	};	END;

Assembler Präfix: *si_*

Datenstrukturenreferenz »Intuition«: *Gadget, IntuitionBase.*

Routinenreferenz Keine.

Beschreibung

Dies ist eine zusätzliche Datenstruktur für »String-Gadgets«. Bei dieser Art von Gadgets kann dem Benutzer eine Zeichenkette angegeben werden, die er beliebig verändern kann. Eine Sonderform ist das Integer-String-Gadget, bei dem Zahlenwerte vom Typ LONG abgefragt werden können. Der Bereich, in dem die Eingabe innerhalb des Gadgets vorgenommen wird, heißt »Container«.

4

- **Buffer 0x0000 (0)** Ein Adreßzeiger auf einen Puffer mit einer Null-terminierten Zeichenkette, die dem Benutzer zur Veränderung angeboten wird. Nachdem der Benutzer die <Return>-Taste gedückt hat, steht hier seine modifizierte Eingabe, der Puffer muß daher groß genug gewählt werden.
- **UndoBuffer 0x0004 (4)** Ein optionaler Adreßzeiger auf einen zweiten Puffer, der eine Kopie der Initialisierungszeichenkette enthält.
- **BufferPos 0x0008 (8)** Die Position innerhalb des Puffers, an der der Cursor beim Aktivieren dieses Gadgets erscheint.
- **MaxChars 0x000A (10)** Die Größe der Puffer in Bytes. **ACHTUNG**: Da die Zeichenkette Null-terminiert ist, muß der Puffer ein Zeichen mehr beinhalten können als maximal eingegeben werden.
- **DispPos 0x000C (12)** Die Position innerhalb des Puffers, an der das erste Zeichen der Zeichenkette ausgegeben wird.
- **UndoPos 0x000E (14)** Hier trägt »Intuition« die aktuelle Position des Cursors in *UndoBuffer* ein.
- **NumChars 0x0010 (16)** Die augenblickliche Anzahl der Zeichen, die sich in der Zeichenkette befinden.
- **DispCount 0x0012 (18)** Die momentane Anzahl der im Container sichtbaren Zeichen der Zeichenkette.
- **CLeft 0x0014 (20)** Hier trägt »Intuition« den Abstand des Containers vom linken Rand des zugehörigen String-Gadgets ein.
- **CTop 0x0016 (22)** Hier trägt »Intuition« den Abstand des Containers vom oberen Rand des zugehörigen String-Gadgets ein.
- **LayerPtr 0x0018 (24)** Hier trägt »Intuition« einen Adreßzeiger auf die zugehörige *Layer*-Datenstruktur ein.
- **LongInt 0x001C (28)** Handelt es sich um ein Integer-String-Gadget, so wird nach dem Drücken der <Return>-Taste durch den Benutzer hier der eingegebene Wert eingetragen.

❑ **AltKeyMap 0x0020 (32)**

Ist in *ActivationFlags* die Flagge *ALTKEYMAP* gesetzt, so steht hier ein Adreßzeiger auf die eigene Keymap, die bei der Eingabe verwendet werden soll.

Window

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez	struct Window {	Window = RECORD
0000	0	struct Window *NextWindow;	nextWindow:WindowPtr;
0004	4	SHORT LeftEdge;	leftEdge:INTEGER;
0006	6	SHORT TopEdge;	topEdge:INTEGER;
0008	8	SHORT Width;	width:INTEGER;
000A	10	SHORT Height;	height:INTEGER;
000C	12	SHORT MouseY;	mouseY:INTEGER;
000E	14	SHORT MouseX;	mouseX:INTEGER;
0010	16	SHORT MinWidth;	minWidth:INTEGER;
0012	18	SHORT MinHeight;	minHeight:INTEGER;
0014	20	USHORT MaxWidth;	maxWidth:INTEGER;
0016	22	USHORT MaxHeight;	maxHeight:INTEGER;
0018	24	ULONG Flags;	flags:WindowFlagSet;
001C	28	struct Menu *MenuStrip;	menuStrip:MenuPtr;
0020	32	UBYTE *Title;	title:ADDRESS;
0024	36	struct Requester *FirstRequest;	firstRequest:RequesterPtr;
0028	40	struct Requester *DMRequest;	dmRequest:RequesterPtr;
002C	44	SHORT ReqCount;	reqCount:INTEGER;
002E	46	struct Screen *WScreen;	wScreen:ScreenPtr;
0032	50	struct RastPort *RPort;	rPort:RastPortPtr;
0036	54	BYTE BorderLeft;	borderLeft:Byte;
0037	55	BYTE BorderTop;	borderTop:Byte;
0038	56	BYTE BorderRight;	borderRight:Byte;
0039	57	BYTE BorderBottom;	borderBottom:Byte;
003A	58	struct RastPort *BorderRPort;	borderRPort:RastPortPtr;
003E	62	struct Gadget *FirstGadget;	firstGadget:GadgetPtr;
0042	66	struct Window *Parent;	parent:WindowPtr;
0046	70	struct Window *Descendant;	descendant:WindowPtr;
004A	74	USHORT *Pointer;	pointer:ADDRESS;
004E	78	BYTE PtrHeight;	ptrHeight:Byte;
004F	79	BYTE PtrWidth;	ptrWidth:[0..16];
0050	80	BYTE XOffset;	xOffset:Byte;
0051	81	BYTE YOffset;	yOffset:Byte;
0052	82	ULONG IDCMPFlags;	idcmpFlags:IDCMPFlagSet;
0056	86	struct MsgPort *UserPort;	userPort:MsgPortPtr;
005A	90	struct MsgPort *WindowPort;	windowPort:MsgPortPtr;

4

Offset		C (intuition/intuition.h)	Modula-2 (Intuition)
Hex	Dez		
005E	94	struct IntuiMessage *MessageKey;	messageKey:IntuiMessagePtr;
0062	98	UBYTE DetailPen;	detailPen:UByte;
0063	99	UBYTE BlockPen;	blockPen:UByte;
0064	100	struct Image *CheckMark;	checkMark:ImagePtr;
0068	104	UBYTE *ScreenTitle;	screenTitle:ADDRESS;
006C	108	SHORT GZZMouseX;	gzzMouseX:INTEGER;
006E	110	SHORT GZZMouseY;	gzzMouseY:INTEGER;
0070	112	SHORT GZZWidth;	gzzWidth:INTEGER;
0072	114	SHORT GZZHeight;	gzzHeight:INTEGER;
0074	116	UBYTE *ExtData;	extData:ADDRESS;
0078	120	BYTE *UserData;	userData:ADDRESS;
007C	124	struct Layer *WLayer;	wLayer:LayerPtr;
0080	128	struct TextFont *IFont;	iFont:TextFontPtr;
0084	132	};	END;

Assembler Präfix: *wd_*

Datenstrukturenreferenz »ConUnit«: *ConUnit.*
»Intuition«: *GListEnv, IntuiMessage, IntuitionBase, Requester, Screen, Window.*

Routinenreferenz »Intuition«: *ActivateGadget, ActivateWindow, AddGadget, AddGList, AutoRequest, BeginRefresh, BuildSysRequest, ClearDMRequest, ClearMenuStrip, ClearPointer, CloseWindow, EndRefresh, EndRequest, FreeSysRequest, ModifyIDCMP, ModifyProp, MoveWindow, NewModifyProp, OffGadget, OffMenu, OnGadget, OnMenu, RefreshGadgets, RefreshGList, RefreshWindowFrame, RemoveGadget, RemoveGList, ReportMouse, Request, SetDMRequest, SetMenuStrip, SetPointer, SetWindowTitles, SizeWindow, ViewPortAddress, WindowLimits, WindowToBack, WindowToFront.*

Beschreibung

Zu jedem geöffneten Fenster gehört eine *Window*-Datenstruktur. In ihr stehen sämtliche Informationen über die Eigenschaften des Fensters.

- **NextWindow 0x0000 (0)** — Sind mehrere Fenster geöffnet, so werden die dazugehörigen *Window*-Datenstrukturen über diesen Adreßzeiger linear verknüpft. Das letzte Element der Liste hat an dieser Stelle den Wert *NULL*.
- **LeftEdge 0x0004 (4)** — Der Abstand des Fensters vom linken Rand des zugehörigen Screens.
- **TopEdge 0x0006 (6)** — Der Abstand des Fensters vom oberen Rand des zugehörigen Screens.
- **Width 0x0008 (8)** — Die Breite des Fensters.
- **Height 0x000A (10)** — Die Höhe des Fensters.
- **MouseY 0x000C (12)** — Ist der Mauszeiger innerhalb dieses Fensters positioniert, so steht hier seine zur linken oberen Ecke relative y-Koordinate.
- **MouseX 0x000E (14)** — Ist der Mauszeiger innerhalb dieses Fensters positioniert, so steht hier seine zur linken oberen Ecke relative x-Koordinate.
- **MinWidth 0x0010 (16)** — Dieser Wert definiert die minimale Breite dieses Fensters. Der Benutzer kann es mit dem Sizing-Gadget minimal bis auf diesen Wert verkleinern.
- **MinHeight 0x0012 (18)** — Dieser Wert definiert die minimale Höhe dieses Fensters. Der Benutzer kann es mit dem Sizing-Gadget minimal bis auf diesen Wert verkleinern.
- **MaxWidth 0x0014 (20)** — Dieser Wert definiert die maximale Breite dieses Fensters. Der Benutzer kann es mit dem Sizing-Gadget maximal bis auf diesen Wert verbreitern.
- **MaxHeight 0x0016 (22)** — Dieser Wert definiert die maximale Höhe dieses Fensters. Der Benutzer kann es mit dem Sizing-Gadget maximal bis auf diesen Wert vergrößern.
- **Flags 0x0018 (24)** — In dieser 4-Byte-Marke stehen die aktuellen WindowFlags. Eine genaue Beschreibung dieser Flags finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.
- **MenuStrip 0x001C (28)** — Ist zu diesem Window eine Menüleiste definiert, so ist dies ein Adreßzeiger auf die erste *Menu*-Datenstruktur.
- **Title 0x0020 (32)** — Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, deren Inhalt auf der Titelleiste dieses Fensters ausgegeben wird.
- **FirstRequest 0x0024 (36)** — Sind in diesem Fenster Requester aktiv, so ist dies ein Adreßzeiger auf die *Requester*-Datenstruktur des zuletzt geöffneten.

- **DMRequest 0x0028 (40)** Sind in diesem Fenster »Doppel-Klick-Requester« aktiv, so ist dies ein Adreßzeiger auf die *Requester*-Datenstruktur des zuletzt geöffneten.
- **ReqCount 0x002C (44)** Hier steht die Anzahl der Requester, die im Moment dieses Window sperren.
- **WScreen 0x002E (46)** Ein Adreßzeiger auf die zu dem Screen, auf dem dieses Window geöffnet wurde, gehörende *Screen*-Datenstruktur.
- **RPort 0x0032 (50)** Ein Adreßzeiger auf die zu diesem Window gehörende *RastPort*-Datenstruktur.
- **BorderLeft 0x0036 (54)** Die Breite des linken Fensterrandes.
- **BorderTop 0x0037 (55)** Die Höhe des oberen Fensterrandes.
- **BorderRight 0x0038 (56)** Die Breite des rechten Fensterrandes.
- **BorderBottom 0x0039 (57)** Die Höhe des unteren Fensterrandes.
- **BorderRPort 0x003A (58)** Ein Adreßzeiger auf eine *RastPort*-Datenstruktur, deren Parameter für das Zeichnen des Fensterrahmens verwendet werden.
- **FirstGadget 0x003E (62)** Ein Adreßzeiger auf die zu dem ersten Gadget in diesem Fenster gehörende *Gadget*-Datenstruktur.
- **Parent 0x0042 (66)** Sind mehrere Fenster geöffnet (egal, auf welchem Screen), so sind alle zugehörigen *Window*-Datenstrukturen über den Adreßzeiger *Parent* rückwärts miteinander verknüpft.
- **Descendant 0x0046 (70)** Sind mehrere Fenster geöffnet (egal, auf welchem Screen), so sind die zugehörigen *Window*-Datenstrukturen über den Adreßzeiger *Descendant* vorwärts miteinander verknüpft.
- **Pointer 0x004A (74)** Ein Adreßzeiger auf die Grafikdaten des zu diesem Window gehörenden Mauszeigers.
- **PtrHeight 0x004E (78)** Die Höhe des Mauszeigers, also die Anzahl der Zeilen, aus der sich die zugehörige Grafik zusammensetzt.
- **PtrWidth 0x004E (78)** Die Breite des Mauszeigers, also die Anzahl der Spalten, aus der sich die zugehörige Grafik zusammensetzt. Diese muß kleiner oder gleich 16 sein.
- **XOffset 0x0050 (80)** Der x-Offset des Hot-Spots in diesem Mauszeiger.
- **YOffset 0x0051 (81)** Der y-Offset des Hot-Spots in diesem Mauszeiger.
- **IDCMPFlags 0x0052 (82)** In dieser 4-Byte-Marke stehen die IDCMP-Flags, die festlegen, über welche Events Systemnachrichten geschickt werden. Eine genaue Beschreibung dieser Flags finden Sie in dem Abschnitt »Wichtige Flags der Intuition-Library«.

- **UserPort 0x0056 (86)** Beim Öffnen des Fenters setzt »Intuition« an diese Stelle einen Adreßzeiger auf eine *MsgPort*-Datenstruktur eines Message-Ports, mit dem Nachrichten empfangen werden können.

- **WindowPort 0x005A (90)** Beim Öffnen des Fensters setzt »Intuition« an diese Stelle einen Adreßzeiger auf eine *MsgPort*-Datenstruktur eines Message-Ports, mit dem Nachrichten gesendet werden können.

- **MessageKey 0x005E (94)** Ein Adreßzeiger auf die zu diesem Window gehörende *IntuiMessage*-Datenstruktur.

- **DetailPen 0x0062 (98)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für die »Details« verwendet wird, also für die Einzelheiten in den Gadgets und die Schrift in der Titelleiste.

- **BlockPen 0x0063 (99)** Die Nummer des Farbregisters, dessen Inhalt als Farbe für »gröbere« Elemente verwendet wird, also für das Füllen der Titelleiste.

- **CheckMark 0x0064 (100)** Ein Adreßzeiger auf die *Image*-Datenstruktur, die den »CheckMark« zu diesem Window definiert.

- **ScreenTitle 0x0068 (104)** Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, deren Inhalt auf der Titelleiste des Screens erscheint.

- **GZZMouseX 0x006C (108)** Ist dies ein GimmeZeroZero-Window und befindet sich der Mauszeiger innerhalb dieses Fensters, so steht hier die momentane x-Koordinate des Mauszeigers relativ zur inneren linken oberen Ecke.

- **GZZMouseY 0x006E (110)** Ist dies ein GimmeZeroZero-Window und befindet sich der Mauszeiger innerhalb dieses Fensters, so steht hier die momentane y-Koordinate des Mauszeigers relativ zur inneren linken oberen Ecke.

- **GZZWidth 0x0070 (112)** Ist dies ein GimmeZeroZero-Window, so steht hier die innere Breite des Fensters, also die Breite ohne den rechten und linken Rand.

- **GZZHeight 0x0072 (114)** Ist dieses Window ein GimmeZeroZero-Window, so steht hier die innere Höhe des Fensters, also die Höhe ohne den oberen und unteren Rand.

- **ExtData 0x0074 (116)** Ein Adreßzeiger für zukünftige Erweiterungen.

- **UserData 0x0078 (120)** Ein frei verwendbarer Adreßzeiger für Zusatzdaten. Damit können Sie jedes Window mit eigenen Daten verknüpfen.

- **WLayer 0x007C (124)** Ein Adreßzeiger auf die *Layer*-Datenstruktur, die bei der Darstellung dieses Fensters benutzt wird.

- **IFont 0x0080 (128)** Ein Adreßzeiger auf eine *TextFont*-Datenstruktur, die den in diesem Window verwendeten Zeichensatz spezifiziert.

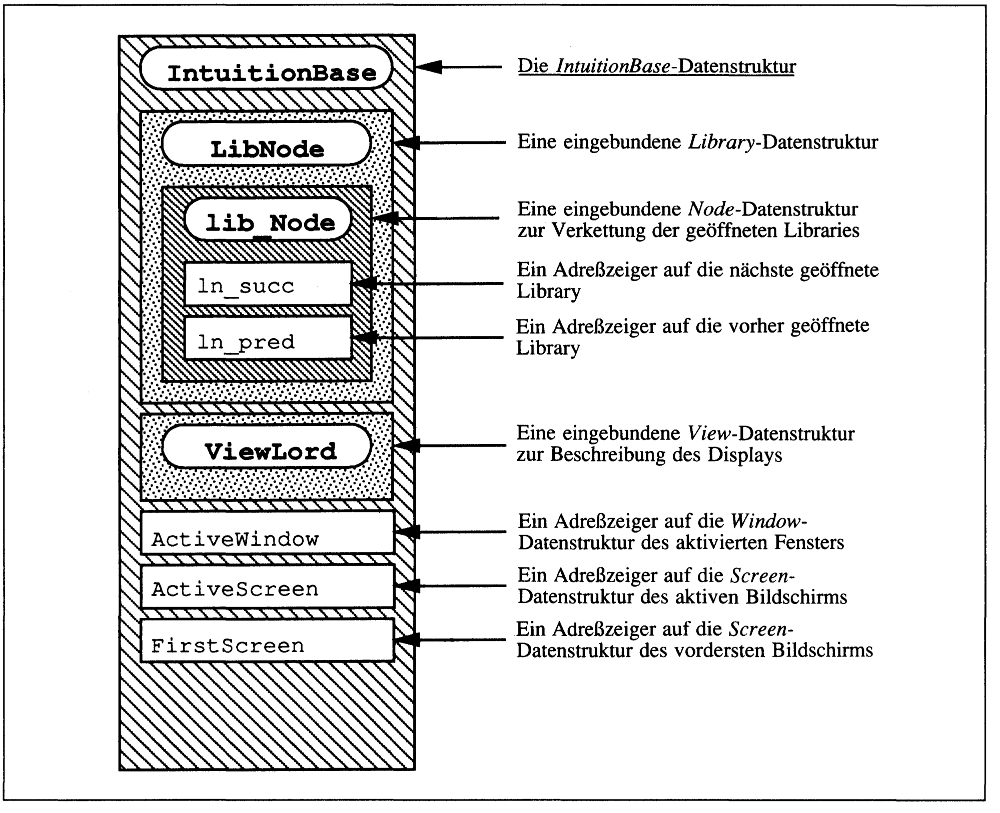


Abb. 4.1: der Aufbau der Intuition-Base-Datenstruktur.

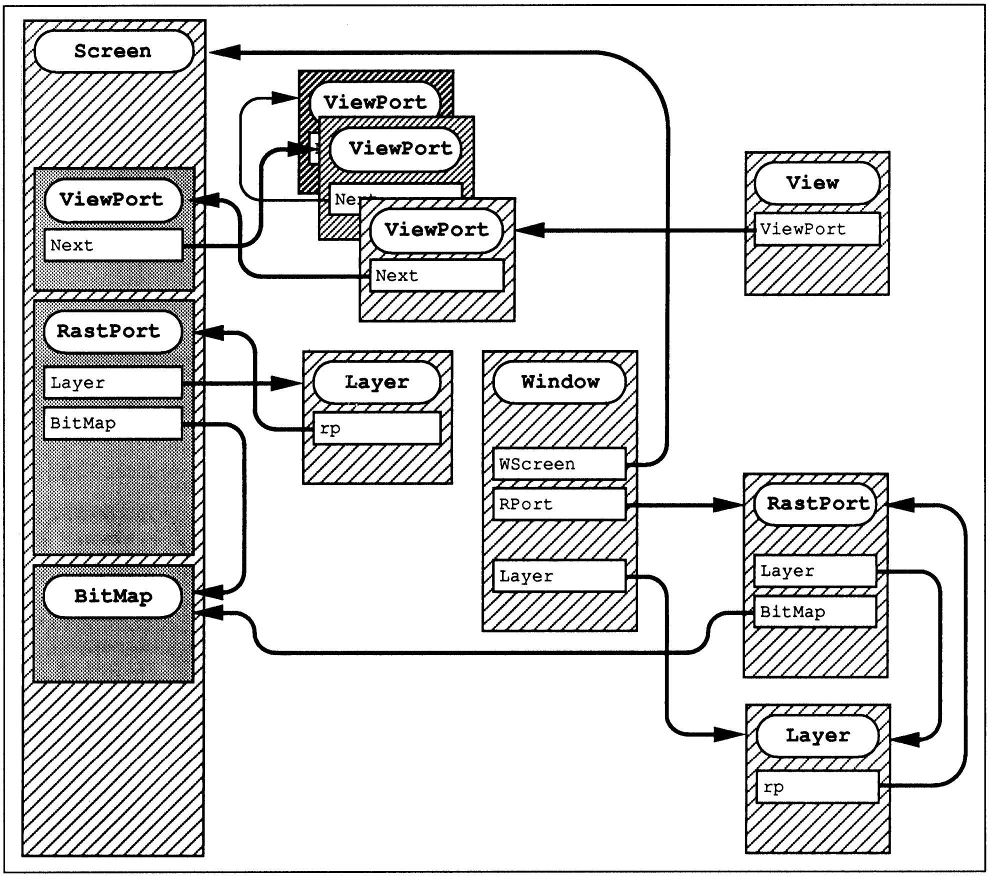


Abb. 4.2: Die Einbindung der Intuition in das Grafiksystem.

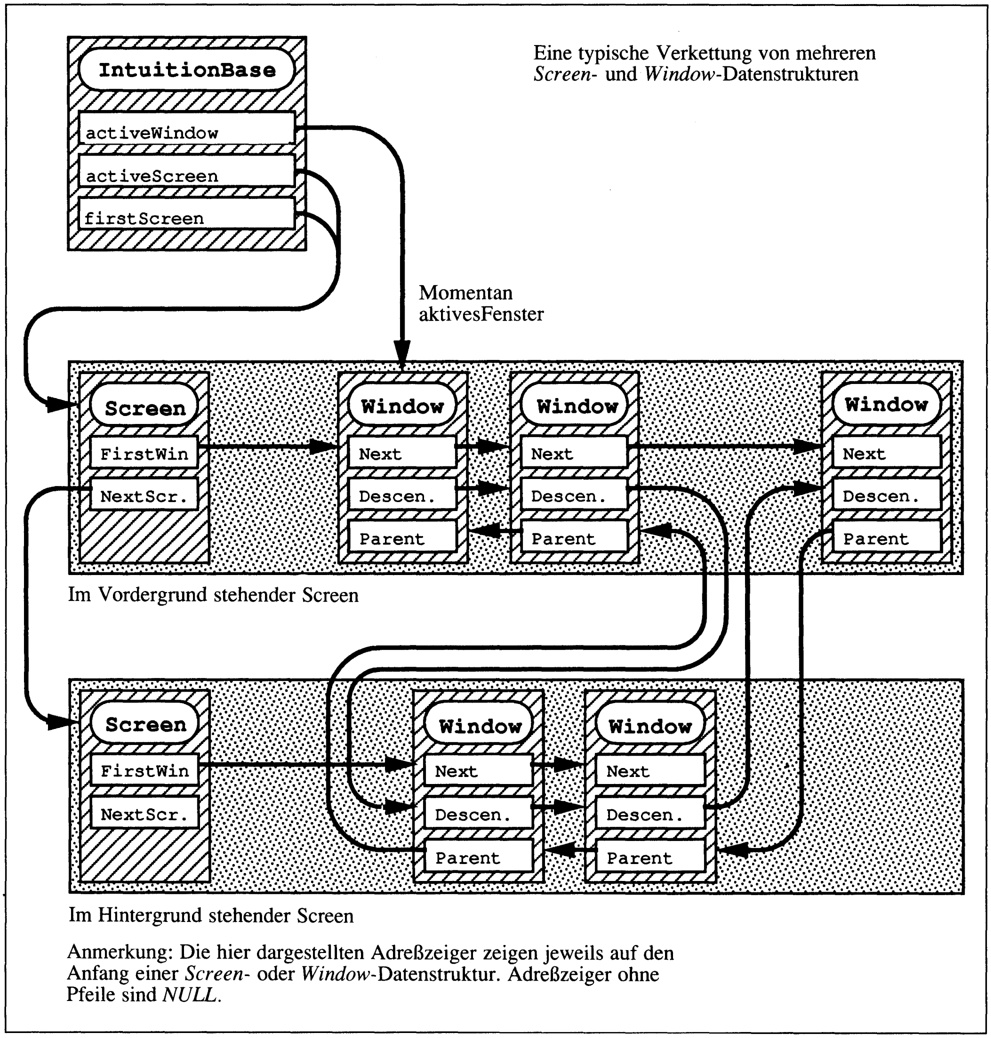


Abb. 4.3: Die Verknüpfung der Screens und Windows.

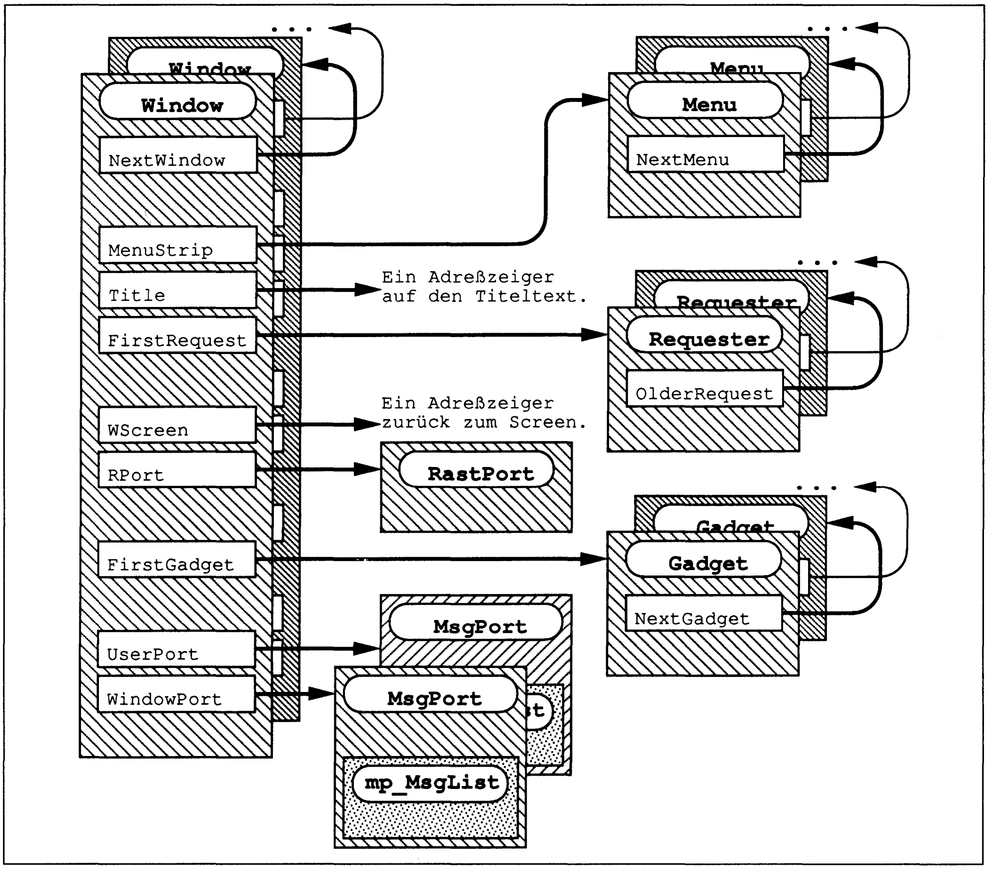


Abb. 4.4.A: Die zu einem Fenster gehörenden Datenstrukturen.

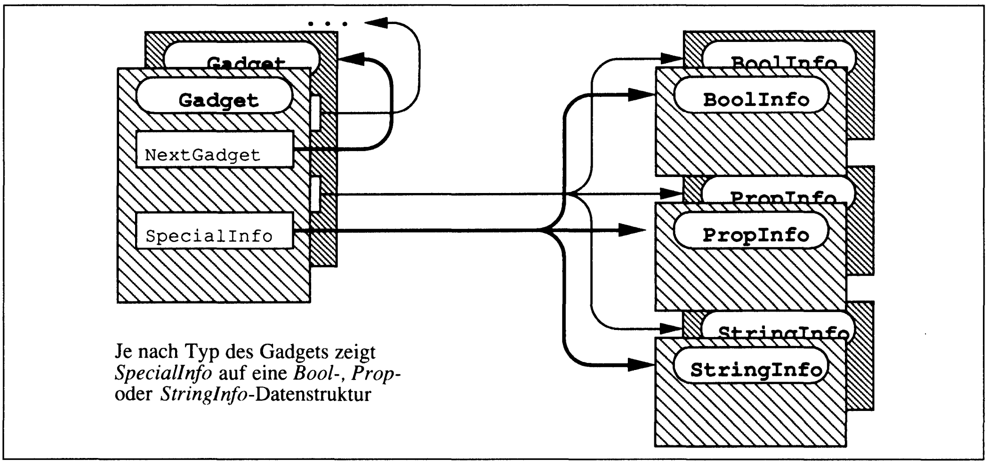


Abb. 4.4.B: . . . wie die der Gadgets . . .

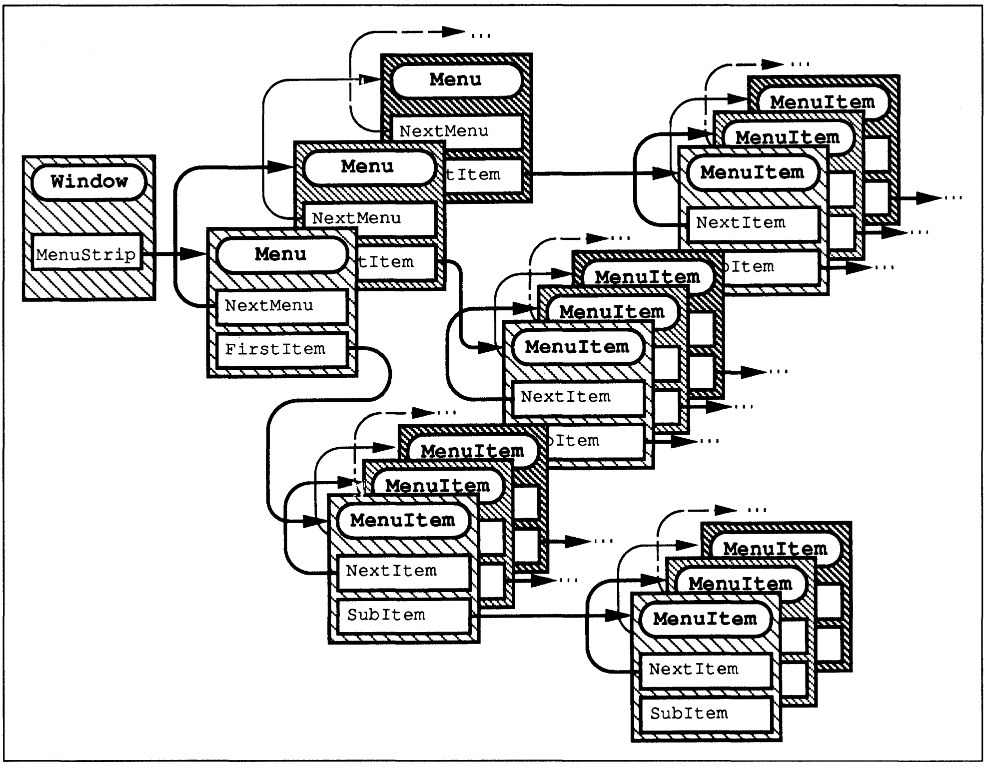


Abb. 4.4.C: . . . und des Menü-Strips.

5 DIE GRAPHICS-LIBRARY

In dieser Library sind alle die Datenstrukturen und Routinen definiert, die zur »grafischen Gestaltung« notwendig sind. Dazu gehören nicht nur diverse Zeichenbefehle, sondern auch Routinen und Datenstrukturen, die die grafische Animation ermöglichen.

5.1 Wichtige Flags der Graphics-Library

DrawModes

(graphics/rastport.h) (Graphics)

C	Modula-2	Hex	Dez	Bedeutung
JAM1	jam1	0000	0	Nur mit der Vordergrundfarbe zeichnen.
JAM2	jam2	0001	1	Mit beiden Farben gleichzeitig zeichnen.
COMPLEMENT	complement	0002	2	Zeichnen durch XOR-Operation der Pixel.
INVERSVIF	inversvid	0004	4	Zeichnen mit Farbinvertierung der Pixel.

Assembler Präfix: *RP_*.

Datenstrukturenreferenz »Graphics«: *RastPort.*
»Intuition«: *Border, IntuiText.*

Routinenreferenz »Graphics«: *SetDrMd.*

Beschreibung

Die einzelnen *DrawModes*-Werte sind für die Festlegung des aktuellen Zeichenmodus innerhalb eines Rastports zuständig.

Wichtig: Unter Modula-2 sind die Zeichenmodi als Aufzählungstyp »*DrawModes = (dm0, complement, inversvid)*« und die eigentliche Bitmaske als Mengentyp *DrawModeSet* definiert. Hier gilt weiterhin: »*jam1=DrawModesSet{}*« und »*jam2=DrawModeSet{dm0}*«!

5

FontFlags

(graphics/text.h) (Graphics)

C	Modula-2	Hex	Dez	Bedeutung
FPF_ROMFONT	romFont	0001	1	Ist der Font im (Pseudo-) ROM?
FPF_DISKFONT	diskFont	0002	2	Ist der Font von Diskette geladen?
FPF_REVPATH	revPath	0004	4	Gilt umgekehrte Schreibrichtung?
FPF_TALLDOT	tallDot	0008	8	Für HIRES & NON-INTERLACE gedacht?
FPF_WIDEDOT	wideDot	0010	16	Für LORES & INTERLACE gedacht?
FPF_PRPORTIONAL	proportional	0020	32	Ist dies ein proportionaler Font?
FPF_DESIGNED	designed	0040	64	Wurde Fontgröße echt geschaffen?
FPF_REMOVED	removed	0080	128	Wurde der Font bereits entfernt?

Datenstrukturenreferenz »Graphics«: *RastPort*, *TextAttr*, *TextFont*.

Routinenreferenz Keine.

Beschreibung

Die einzelnen *FontFlags*-Werte dienen zur Angabe wichtiger Design- und Laufzeit-Charakteristika eines Textfonts.

Wichtig: Unter Modula-2 sind die *FontFlags* als Aufzählungstyp »*FontFlags = (romFont, diskFont, revPath, tallDot, wideDot, proportional, designed, removed)*« und die eigentliche Bitmaske als Mengentyp *FontFlagSet* definiert.

FontStyles

(graphics/text.h) (Graphics)

C	Modula-2	Hex	Dez	Bedeutung
FS_NORMAL	normalFont	0000	0	Normaler Text, d.h. nichts gesetzt?
FSF_UNDERLINED	underlined	0001	1	Unterstrichene Zeichenausgabe?

C	Modula-2	Hex	Dez	Bedeutung
FSF_BOLD	bold	0002	2	Fette Zeichenausgabe?
FSF_ITALIC	italic	0004	4	Kursive Zeichenausgabe?
FSF_EXTENDED	extended	0008	8	Breitere Zeichen als normal?

Datenstrukturenreferenz »Graphics«: *RastPort*, *TextAttr*, *TextFont*.

Routinenreferenz »Graphics«: *AskSoftStyle*, *SetSoftStyle*.

Beschreibung

Die einzelnen *FontStyles*-Werte dienen zur Auswahl und Angabe des (eventuell algorithmisch zu erzeugenden) Schrifttyps der Zeichen eines Textfonts.

Wichtig: Unter Modula-2 sind die Schrifttypen als Aufzählungstyp »*FontStyles = (underlined, bold, italic, extended)*« und die eigentliche Bitmaske als Mengentyp *FontStyleSet* definiert. Hier gilt weiterhin: »*normalFont=FontStyleSet{}*«!

5

ViewModes

(graphics/view.h) (Graphics)

C	Modula-2	Hex	Dez	Bedeutung
GENLOCK_VIDEO	genlocVideo	0002	2	Video-Genlock vorhanden?
LACE	lace	0004	4	Interlace-Modus ausgewählt?
PFBA	pfba	0040	64	Playfield 2 vor Playfield 1?
EXTRA_HALFBRITE	extraHalfbrite	0080	128	Extra-Halfbrite-Modus gewählt?
GENLOCK_AUDIO	genlocAudio	0100	256	Audio-Genlock vorhanden?
DUALPF	dualpf	0400	1024	Dual-Playfield-Modus gewählt?
HAM	ham	0800	2048	Hold-And-Modify-Modus gewählt?
VP_HIDE	vpHide	2000	8192	Viewport vollkommen verdeckt?
SPRITES	sprites	4000	16384	Werden hier Sprites benutzt?
HIRES	hires	8000	32768	High-Resolution-Modus gewählt?

Datenstrukturenreferenz »Graphics«: *View*, *ViewPort*.
»Intuition«: *NewScreen*, (*Screen*).

Routinenreferenz Keine.

Beschreibung

Die einzelnen *ViewModes*-Werte dienen der Beschreibung des tatsächlichen Darstellungsmodus und anderer Charakteristika des Gesamtdisplays oder eines Viewports.

Wichtig: Unter Modula-2 sind die *ViewModes* als Aufzählungstyp »*ViewModes = (vm0, genlocVideo, lace, vm3, vm5, pfba, extraHalfbrite, genlcAudio, vm9, dualpf, ham, vm12, vpHide, sprites, hires)*« und die eigentliche Bitmaske als Mengentyp *ViewModeSet* definiert.

5

5.2 Die Datenstrukturen der Graphics-Library

AnimComp

Offset Hex	C Dez	(graphics/gels.h)	Modula-2 (Graphics)
		struct AnimComp	AnimComp =
		{	RECORD
0000	0	WORD Flags;	flags:INTEGER;
0002	2	WORD Timer;	timer:INTEGER;
0004	4	WORD TimeSet;	timeSet:INTEGER;
0006	6	struct AnimComp *NextComp;	nextComp:AnimCompPtr;
000A	10	struct AnimComp *PrevComp;	prevComp:AnimCompPtr;
000E	14	struct AnimComp *NextSeq;	nextSeq:AnimCompPtr;
0012	18	struct AnimComp *PrevSeq;	prevSeq:AnimCompPtr;
0016	22	WORD (*AnimCRoutine)();	animCRoutine:ADDRESS;
001A	26	WORD YTrans;	yTrans:INTEGER;
001C	28	WORD XTrans;	xTrans:INTEGER;
001E	30	struct AnimOb *HeadOb;	headOb:AnimObPtr;
0022	34	struct Bob *AnimBob;	animBob:BobPtr;
0026	38	};	END;

Assembler Präfix: *ac_*
Syntax: Strukturname = *AC*

Datenstrukturenreferenz »Graphics«: *AnimComp*, *AnimOb*, *Bob*.

Routinenreferenz Keine.

Beschreibung

Die *AnimComp*-Datenstruktur wird vom Animationssystem zur Verwaltung der unterschiedlichen Einzelkomponenten eines ganzen Animationsobjekts verwendet (siehe *AnimOb*-Datenstruktur). Jeder

AnimComp-Datenstruktur ist dabei genau ein Bob zugeordnet, das sowohl eine Einzelkomponente insgesamt als auch nur eine Momentaufnahme der Bewegungssequenz einer Einzelkomponente darstellen kann.

❑ **Flags 0x0000 (0)**

Eine 2-Byte-Marke, deren niederwertigstes Bit gesetzt ist, falls diese Einzelkomponente nur eine Momentaufnahme einer Bewegungssequenz ist. Zur Benutzung sind hier folgende Konstanten vordefiniert:

C	Modula-2	Hex	Dez	Bedeutung
RINGTRIGGER	ringTrigger	0001	1	Handelt es sich um Momentaufnahme?
ANIMHALF	animHalf	0020	32	Reserviert für internen Systemgebrauch.

5

❑ **Timer 0x0002 (2)**

Eine positive Ganzzahl, deren Wert angibt, wie lange diese Einzelkomponente (z.B. als Teil einer Bewegungssequenz) noch aktiv bleiben soll. Gemessen wird hierbei in Aufrufen der »Graphics«-Systemroutine *Animate*. Ist dieser Wert permanent gleich null, so bleibt diese Einzelkomponente dauernd aktiv!

❑ **TimeSet 0x0004 (4)**

Eine positive Ganzzahl, deren Wert bei jeder Aktivierung der Einzelkomponente in die *Timer*-Variable ihrer *AnimComp*-Datenstruktur eingetragen wird.

❑ **NextComp 0x0006 (6)**

Ein Adreßzeiger auf die *AnimComp*-Datenstruktur, die der nächsten Einzelkomponente des betreffenden Animationsobjekts zugeordnet ist. Über die *NextComp*-Zeiger werden sämtliche Einzelkomponenten eines Animationsobjekts vorwärts verkettet.

❑ **PrevComp 0x000A (10)**

Ein Adreßzeiger auf die *AnimComp*-Datenstruktur, die der vorangehenden Einzelkomponente des betreffenden Animationsobjekts zugeordnet ist. Über die *NextComp*-Zeiger werden sämtliche Einzelkomponenten eines Animationsobjekts rückwärts verkettet.

❑ **NextSeq 0x000E (14)**

Ein Adreßzeiger auf die *AnimComp*-Datenstruktur der nächsten Momentaufnahme innerhalb der Bewegungssequenz, zu der diese Einzelkomponente gehört. Über die *NextSeq*-Zeiger werden sämtliche Momentaufnahmen einer Bewegungssequenz vorwärts verkettet.

❑ **PrevSeq 0x0012 (18)**

Ein Adreßzeiger auf die *AnimComp*-Datenstruktur der nächsten Momentaufnahme innerhalb der Bewegungssequenz, zu der diese Einzelkomponente gehört. Über die *PrevSeq*-Zeiger werden sämtliche Momentaufnahmen einer Bewegungssequenz vorwärts verkettet.

5

- **AnimCRoutine 0x0016 (22)** Die Adresse des Codebereichs einer Routine, die nach jedem Animationsschritt zur weiteren Bearbeitung dieser Einzelkomponente aufgerufen wird.
- **YTrans 0x001A (26)** Eine Ganzzahl, deren Wert die y-Koordinate der linken oberen Ecke dieser Einzelkomponente relativ zum Registrationspunkt des übergeordneten Animationsobjekts angibt.
- **XTrans 0x001C (28)** Eine Ganzzahl, deren Wert die x-Koordinate der linken oberen Ecke dieser Einzelkomponente relativ zum Registrationspunkt des übergeordneten Animationsobjekts angibt.
- **HeadOb 0x001E (30)** Ein Adreßzeiger auf die *AnimOb*-Datenstruktur des Animationsobjekts, dem diese Einzelkomponente zugeordnet ist.
- **AnimBob 0x0022 (34)** Ein Adreßzeiger auf die *Bob*-Datenstruktur des Bobs, das speziell dieser Einzelkomponente zugeordnet ist.

AnimOb

Offset		C (graphics/gels.h)	Modula-2 (Graphics)
Hex	**Dez**	struct AnimOb	AnimOb =
		{	RECORD
0000	0	struct AnimOb *NextOb;	nextOb:AnimObPtr;
0004	4	struct AnimOb *PrevOb;	prevOb:AnimObPtr;
0008	8	LONG Clock;	clock:LONGINT;
000C	12	WORD AnOldY;	anOldY:INTEGER;
000E	14	WORD AnOldX;	anOldX:INTEGER;
0010	16	WORD AnY;	anY:INTEGER;
0012	18	WORD AnX;	anX:INTEGER;
0014	20	WORD YVel;	yVel:INTEGER;
0016	22	WORD XVel;	xVel:INTEGER;
0018	24	WORD YAccel;	yAccel:INTEGER;
001A	26	WORD XAccel;	xAccel:INTEGER;
001C	28	WORD RingYTrans;	ringYTrans:INTEGER;
001E	30	WORD RingXTrans;	ringXTrans:INTEGER;
0020	32	WORD (*AnimORoutine)();	animORoutine:ADDRESS;
0024	36	struct AnimComp *HeadComp;	headComp:AnimCompPtr;
0028	40	AUserStuff AUserExt;	
002A	42	};	END;

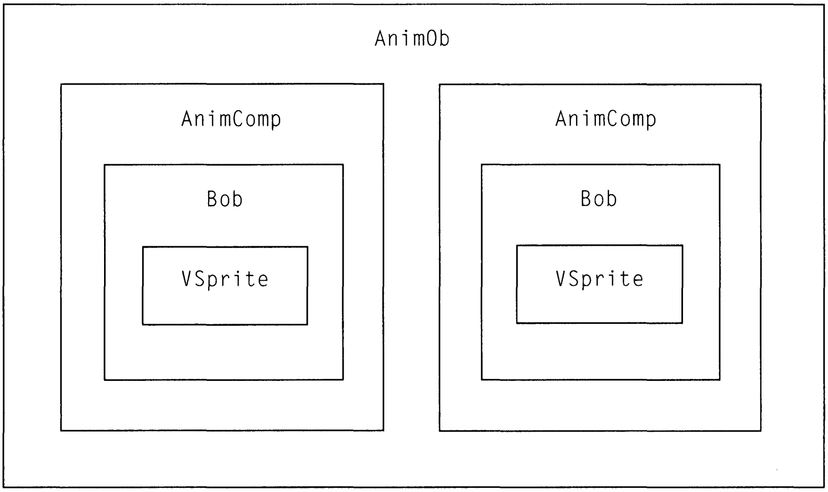
Assembler Präfix: *ao_*
Syntax: Strukturname = *AO*

Datenstrukturenreferenz »Graphics«: *AnimComp*, *AnimOb*.

Routinenreferenz »Graphics«: *AddAnimOb*, *FreeGBuffers*, *GetGBuffers*, *InitGMasks*.

Beschreibung

Die *AnimOb*-Datenstruktur dient der Systemsoftware zur Verwaltung ganzer Animationsobjekte. Insgesamt kennt das Animationssystem des Amigas folgende Animationselemente: Hardwaresprites, virtuelle Sprites, Bobs, *AnimComp*-Einzelkomponenten und *AnimOb*-Animationsobjekte. Dabei erweisen sich die Hardwaresprites in dieser Aufstellung als Ausnahme, da sie im Gegensatz zu den übrigen Animationselementen direkt von der Hardware gesteuert werden. Die von der Systemsoftware gesteuerten Animationselemente bauen in der angegebenen Reihenfolge aufeinander auf: Ein Bob etwa besteht zur Hälfte aus einem virtuellen Sprite, eine *AnimComp*-Einzelkomponente enthält ein oder mehrere Bobs und ein Animationsobjekt setzt sich aus mehreren Einzelkomponenten zusammen! Alles in allem sieht der hierarchische Aufbau eines Animationsobjekts also folgendermaßen aus:



5

- **NextOb 0x0000 (0)** — Ein Adreßzeiger auf die *AnimOb*-Datenstruktur des nächsten Animationsobjekts, das in der betreffenden Animationsliste zu bewegen ist. Über die *NextOb*-Zeiger werden sämtliche Animationsobjekte einer Animationsliste vorwärts verkettet.
- **PrevOb 0x0004 (4)** — Ein Adreßzeiger auf die *AnimOb*-Datenstruktur des vorangehenden Animationsobjekts, das in der betreffenden Animationsliste zu bewegen ist. Über die *PrevOb*-Zeiger werden sämtliche Animationsobjekte einer Animationsliste rückwärts verkettet.
- **Clock 0x0008 (8)** — Eine lange Ganzzahl, deren Wert angibt, wie oft das Animationsobjekt schon mittels Aufrufen der »Graphics«-Systemroutine *Animate* bewegt wurde.
- **AnOldY 0x000C (12)** — Eine Ganzzahl, deren Wert die y-Koordinate der letzten Position des Registrationspunkts dieses Animationsobjekts angibt.

- **AnOldX 0x000E (14)** Eine Ganzzahl, deren Wert die x-Koordinate der letzten Position des Registrationspunkts dieses Animationsobjekts angibt.

- **AnY 0x0010 (16)** Eine Ganzzahl, deren Wert die y-Koordinate der aktuellen Position des Registrationspunkts dieses Animationsobjekts festlegt.

- **AnX 0x0012 (18)** Eine Ganzzahl, deren Wert die x-Koordinate der aktuellen Position des Registrationspunkts dieses Animationsobjekts festlegt.

- **YVel 0x0014 (20)** Eine Ganzzahl, deren Wert die vertikale Geschwindigkeit des gesamten Animationsobjekts in 1/64 Pixeln pro Bewegungsschritt angibt.

- **XVel 0x0016 (22)** Eine Ganzzahl, deren Wert die horizontale Geschwindigkeit des gesamten Animationsobjekts in 1/64 Pixeln pro Bewegungsschritt angibt.

- **YAccel 0x0018 (24)** Eine Ganzzahl, deren Wert die vertikale Beschleunigung des gesamten Animationsobjekts in 1/64 Pixeln pro Bewegungsschritt angibt.

- **XAccel 0x001A (26)** Eine Ganzzahl, deren Wert die horizontale Beschleunigung des gesamten Animationsobjekts in 1/64 Pixeln pro Bewegungsschritt angibt.

- **RingYTrans 0x001C (28)** Eine Ganzzahl, deren Wert den vertikalen Versatz des Registrationspunkts dieses Animationsobjekts angibt, der zu seiner aktuellen Position hinzuaddiert wird, falls ein ganzer Ablauf der Bewegungssequenz einer Teilkomponente durchgeführt wurde. Die Addition findet selbstverständlich nur bei gesetztem *RINGTRIGGER*-Bit der entsprechenden Einzelkomponente statt.

- **RingXTrans 0x001E (30)** Eine Ganzzahl, deren Wert den vertikalen Versatz des Registrationspunkts dieses Animationsobjekts angibt, der zu seiner aktuellen Position hinzuaddiert wird, falls ein ganzer Ablauf der Bewegungssequenz einer Teilkomponente durchgeführt wurde. Die Addition findet selbstverständlich nur bei gesetztem *RINGTRIGGER*-Bit der entsprechenden Einzelkomponente statt.

- **AnimORoutine 0x0020 (32)** Die Adresse des Codebereichs einer Routine, die nach jedem Animationsschritt zur weiteren Bearbeitung dieses Animationsobjekts aufgerufen wird.

- **HeadComp 0x0024 (36)** Ein Adreßzeiger auf die *AnimComp*-Datenstruktur der Einzelkomponente, die den Anfang der Liste von Einzelkomponenten des Animationsobjekts darstellt.

❑ **AUserExt 0x0028 (40)**

Eine vom Benutzer völlig frei definierbare Erweiterung der *AnimOb*-Datenstruktur zur Aufnahme weiterer Statusinformationen. Zu diesem Zweck ist der Typ *AUserStuff* im Benutzerprogramm entsprechend zu deklarieren. (Wichtig: Diese Erweiterung ist nur unter C, nicht jedoch unter Modula-2 verfügbar!)

AreaInfo

Offset		C (graphics/rastport.h)	Modula-2 (Graphics)
Hex	**Dez**	struct AreaInfo	AreaInfo =
		{	RECORD
0000	0	SHORT *VctrTbl;	vctrTbl:ADDRESS;
0004	4	SHORT *VctrPtr;	vctrPtr:ADDRESS;
0008	8	BYTE *FlagTbl;	flagTbl:ADDRESS;
000C	12	BYTE *FlagPtr;	flagPtr:ADDRESS;
0010	16	SHORT Count;	count:INTEGER;
0012	18	SHORT MaxCount;	maxCount:INTEGER;
0014	20	SHORT FirstX;	firstX:INTEGER;
0016	22	SHORT FirstY;	firstY:INTEGER;
0018	24	};	END;

Assembler Präfix: *ai_*

Datenstrukturenreferenz »Graphics«: *RastPort.*

Routinenreferenz »Graphics«: *InitArea.*

Beschreibung

Die *AreaInfo*-Datenstruktur wird zur Speicherung und Verwaltung eines (zu füllenden) Polygons benutzt und von den verschiedenen *Areaxxx*-Routinen benötigt.

❑ **VctrTbl 0x0000 (0)**

Ein Adreßzeiger auf den Anfang der Tabelle der Polygoneckpunkte, die punktweise organisiert ist. Für jeden Eckpunkt werden dort seine x- und y-Koordinaten direkt aufeinanderfolgend abgelegt. Insgesamt werden pro Punkt 5 Byte Speicherplatz in der Tabelle benötigt, da neben den eigentlichen Koordinaten am Ende der Tabelle noch diverse Flaggen gespeichert sind.

❑ **VctrPtr 0x0004 (4)**

Ein Adreßzeiger auf das Koordinatenpaar des aktuellen Eckpunkts in der *VctrTbl*-Tabelle.

❑ **FlagTbl 0x0008 (8)**

Ein Adreßzeiger auf den Anfang der punktweise organisierten Tabelle von Vektorflags für das Polygon.

- **FlagPtr 0x000C (12)** — Ein Adreßzeiger auf den Anfang der punktweise organisierten Tabelle von Füllflags für das Polygon. Diese Füllflags dienen der Speicherung von Clipping-Informationen.
- **Count 0x0010 (16)** — Eine positive Ganzzahl, deren Wert die momentan in der *VctrTbl*-Tabelle gespeicherte Anzahl von Eckpunkten des Polygons angibt.
- **MaxCount 0x0012 (18)** — Eine positive Ganzzahl, deren Wert die maximal mögliche Anzahl von Eckpunkten festlegt, die in der *VctrTbl*-Tabelle Platz finden.
- **FirstX 0x0014 (20)** — Eine Ganzzahl, deren Wert der x-Koordinate des ersten Eckpunkts des Polygons entspricht.
- **FirstY 0x0016 (22)** — Eine Ganzzahl, deren Wert der y-Koordinate des ersten Eckpunkts des Polygons entspricht.

BitMap

Offset		C (graphics/gfx.h)	Modula-2 (Graphics)
Hex	**Dez**	struct BitMap	BitMap =
		{	RECORD
0000	0	UWORD BytesPerRow;	bytesPerRow:CARDINAL;
0002	2	UWORD Rows;	rows:CARDINAL;
0004	4	UBYTE Flags;	flags:UByte;
0005	5	UBYTE Depth;	depth:UByte;
0006	6	UWORD pad;	pad:CARDINAL;
0008	8	PLANEPTR Planes[8];	planes:ARRAY [0..7] OF ADDRESS;
0028	40	};	END;

Assembler — Präfix: *bm_*
Syntax: *pad —> bm_Pad*

Datenstrukturenreferenz — »Graphics«: *ClipRect, Layer, RasInfo, RastPort.*
»Intuition«: *IntuitionBase, NewScreen, NewWindow, Requester, Screen.*

Routinenreferenz — »Graphics«: *BltBitMap, BltBitMapRastPort, BltMaskBitMapRastPort, InitBitMap.*
»Layers«:< *CreateBehindLayer, CreateUpfront Layer.*

Beschreibung

Die *BitMap*-Datenstruktur dient zur Speicherung der grundlegenden Statusinformationen für den bzw. die Displayspeicherbereiche eines beliebigen Displayelements. Neben der Höhe, Breite und Tiefe sind hier ebenfalls die Zeiger auf die einzelnen Bitplanes vorhanden.

- ❑ **BytesPerRow 0x0000 (0)** Eine positive Ganzzahl, deren Wert die Anzahl von Speicherbytes je Rasterzeile für die einzelnen Bitplanes angibt.
- ❑ **Rows 0x0002 (2)** Eine positive Ganzzahl, deren Wert die Anzahl von Rasterzeilen und damit die Höhe der einzelnen Bitplanes angibt.
- ❑ **Flags 0x0004 (4)** Eine nur für den Systemgebrauch bestimmte Marke.
- ❑ **Depth 0x0005 (5)** Eine positive Ganzzahl, deren Wert die Tiefe der Bitmap angibt. Diese Tiefe entspricht der Anzahl vorhandener Bitplanes.
- ❑ **pad 0x0006 (6)** Eine nur für den Systemgebrauch bzw. spätere Erweiterungen bestimmte Füllkomponente.
- ❑ **Planes 0x0008 (8)** Ein Feld von acht Adreßzeigern auf die Anfänge der einzelnen Bitplanes.

5

Bob

Offset		C (graphics/gels.h)	Modula-2 (Graphics)
Hex	Dez	struct Bob	Bob =
		{	RECORD
0000	0	WORD Flags;	flags:BobFlagSet;
0002	2	WORD *SaveBuffer;	saveBuffer:ADDRESS;
0006	6	WORD *ImageShadow;	imageShadow:ADDRESS;
000A	10	struct Bob *Before;	before:BobPtr;
000C	14	struct Bob *After;	after:BobPtr;
0012	18	struct VSprite *BobVSprite;	bobVSprite:VSpritePtr;
0016	22	struct AnimComp *BobComp;	bobComp:AnimCompPtr;
001A	26	struct DBufPacket *DBuffer;	dBuffer:DBufPacketPtr;
001E	30	BUserStuff BUserExt;	
0020	32	};	END;

Assembler Präfix: *bob_*
Syntax: Strukturname = *BOB*
Flags —> bob_BobFlags

Datenstrukturenreferenz »Graphics«: *AnimComp*, *Bob*, *VSprite*.

Routinenreferenz »GfxMacros«: *RemBob*.
»Graphics«: *AddBob*, *RemIBob*.

Beschreibung

Die *Bob*-Datenstruktur wird von den Animationsroutinen zur Speicherung der Daten verwendet, die neben der ohnehin benutzten *VSprite*-Datenstruktur zur Verwaltung eines Bobs = »blitter object« noch zusätzlich benötigt werden. Die Notwendigkeit einer eigenen (Zusatz-)Datenstruktur für Bobs ergibt sich

daraus, daß sich Bobs und Sprites im Hinblick auf die Erzeugung ihres Ausgabebilds auf dem Monitor grundlegend unterscheiden: Sprites sind echte Hardwareobjekte, die völlig unabhängig von den Bitplanes der aktuellen Playfields sichtbar gemacht werden. Für das Ausgabebild eines Sprites ist somit die spezielle Spritehardware zuständig. Bobs dagegen sind Playfield-Objekte. Zur Erzeugung des Ausgabebilds eines Bobs muß dessen Image in die Bitplanes des entsprechenden Playfields gezeichnet werden. Bei der Bewegung eines Bobs ist deshalb der Playfield-Hintergrund ständigen Speicher- und Restaurierungsaktionen zu unterziehen. Im Falle eines Sprites sind solche Maßnahmen nicht vonnöten, weil das Sprite den Zustand der Playfields überhaupt nicht verändert.

❑ **Flags 0x0000 (0)**

Eine 2-Byte-Marke, deren einzelne Bits Statusinformationen über den aktuellen Zustand und Darstellungsmodus des Bobs enthalten. Zur Benutzung sind hier folgende Konstanten vordefiniert, von denen allerdings nur die beiden ersten und das fünfte solche Bits beschreiben, die vom Benutzer verändert werden können. Die restlichen Bits dürfen einzig und allein gelesen werden.

5

C	Modula-2	Hex	Dez	Bedeutung
SAVEBOB	saveBob	0001	1	Altes Bobimage nicht löschen?
BOBISCOMP	bobIsComp	0002	2	Ist das Bob Teil eines AnimComp?
BWAITING	bWaiting	0100	256	Wartet das Bob auf seine Darstellung?
BDRAWN	bDrawn	0200	512	Ist das Bob bereits gezeichnet?
BOBSAWAY	bobsAway	0400	1024	Soll das Bob entfernt werden?
BOBNIX	bobNix	0800	2048	Ist das Bob schon ganz entfernt?
SAVEPRESERVE	savePreserve	1000	4096	Ist der Hintergrund zu speichern?
OUTSTEP	outStep	2000	8192	Ist der Hintergrund zu löschen?

Die letzten beiden Bits *SAVEPRESERVE* und *OUTSTEP* werden von den Animationsroutinen im doppelt gepufferten Darstellungsmodus benötigt. Das *SAVEBOB*-Bit gibt an, ob das an der alten Position gezeichnete Bobimage bei der Bewegung gelöscht werden soll oder nicht: Ist das *SAVEBOB*-Bit gesetzt, so wird das alte Bobimage nicht gelöscht. Auf diese Art und Weise kann man ein Bob sehr einfach und elegant als Zeichenpinsel benutzen.

❑ **SaveBuffer 0x0002 (2)**

Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, in dem die bei der Darstellung des Bobs überschriebenen Hintergrunddaten des Playfields gespeichert werden. Die Größe dieses Speicherbereichs muß mindestens *Height*Width*Depth*/8 Byte betragen, wobei Dimensionswerte aus der *VSprite*-Datenstruktur des Bobs stammen.

❑ **ImageShadow 0x0006 (6)**

Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der die Größe einer Bitplane des Bobs haben muß und somit mindestens *Height**Width/8* Byte umfassen muß. Die Dimensionswerte entstammen dabei der *VSprite*-Datenstruktur des Bobs. Der Imageshadow wird beim Zeichnen des Bobs im Rahmen der Bitplane-Maskierung des Playfields benötigt (siehe *PlanePick-* und *PlaneOnOff*-Masken der *VSprite*-Datenstruktur).

❑ **Before 0x000A (10)**

Ein Adreßzeiger auf die *Bob*-Datenstruktur des Bobs in der GEL-Liste, das direkt vor diesem Bob gezeichnet werden soll.

❑ **After 0x000C (14)**

Ein Adreßzeiger auf die *Bob*-Datenstruktur des Bobs in der GEL-Liste, nach dem dieses Bob gezeichnet werden soll.

5

❑ **BobVSprite 0x0012 (18)**

Ein Adreßzeiger auf die zu diesem Bob gehörige *VSprite*-Datenstruktur, in der die restlichen Statusdaten des Bobs gespeichert sind.

❑ **BobComp 0x0016 (22)**

Ein Adreßzeiger auf die *AnimComp*-Datenstruktur des Verbunds von Animationsobjekten, zu dem dieses Bob gehört. Der Wert dieses Adreßzeigers wird von den Animationsroutinen nur dann beachtet, wenn das *BOBISCOMP*-Bit der *Flags*-Marke in der *Bob*-Datenstruktur des Bobs gesetzt ist.

❑ **DBuffer 0x001A (26)**

Ein Adreßzeiger auf eine *DBufPacket*-Datenstruktur, die im doppelt gepufferten Darstellungsmodus benötigt wird. Der doppelt gepufferte Darstellungsmodus wird durch einen Wert dieses Adreßzeigers ungleich *NULL* bzw. *NIL* ausgewählt. Falls normaler Darstellungsmodus erwünscht ist, muß der *DBuffer*-Zeiger somit mit dem Wert *NULL* bzw. *NIL* initialisiert werden.

Wichtig: Falls ein Bob einer GEL-Liste doppelt gepuffert werden soll, dann sind auch alle übrigen Bobs der GEL-Liste doppelt zu puffern!

❑ **BUserExt 0x001E (30)**

Eine vom Benutzer völlig frei definierbare Erweiterung der *Bob*-Datenstruktur zur Aufnahme weiterer Statusinformationen. Zu diesem Zweck ist der Typ *BUserStuff* im Benutzerprogramm entsprechend zu deklarieren. (Wichtig: Diese Erweiterung ist nur unter C, nicht jedoch unter Modula-2 verfügbar!)

ClipRect

Offset		C (graphics/clip.h)	Modula-2 (Graphics)
Hex	Dez	struct ClipRect	ClipRect =
		{	RECORD
0000	0	struct ClipRect *Next;	next:ClipRectPtr;
0004	4	struct ClipRect *prev;	prev:ClipRectPtr;
0008	8	struct Layer *lobs;	lobs:LayerPtr;
000C	12	struct BitMap *BitMap;	bitMap:BitMapPtr;
0010	16	struct Rectangle bounds;	bounds:Rectangle;
0018	24	struct ClipRect *_p1;	p1:ClipRectPtr;
001C	28	struct ClipRect *_p2;	p2:ClipRectPtr;
0020	32	LONG reserved;	reserved:LONGINT;
0024	36	LONG Flags;	flags:LONGINT;
0028	40	};	END;

Assembler	Präfix:	*cr_*
	Syntax:	*struct Rectangle bounds —> WORD cr_MinX, cr_MinY, cr_MaxX, cr_MaxY*
Datenstrukturenreferenz	»Graphics«:	*ClipRect, Layer.*
	»Intuition«:	*IntuitionBase.*
Routinenreferenz	»Layers«:	*SwapBitsRastPortClipRect.*

Beschreibung

Die *ClipRect*-Datenstruktur dient der internen Repräsentation eines rechteckigen Clipping-Bereichs, der den Grafikroutinen beim Zugriff auf einen Layer angibt, in welche Bitmap gezeichnet werden soll. Mehrere Listen von *ClipRect*-Datenstrukturen werden vom System zur Kennzeichnung und Verwaltung sämtlicher Clipping-und Damage-Bereiche eines Layers verwendet.

❑ **Next 0x0000 (0)**

Ein Adreßzeiger auf die *ClipRect*-Datenstruktur des nachfolgenden Clipping-Bereichs in der Clipping-Bereichsliste. Über die NextZeiger werden sämtliche Clipping-Bereiche einer Clipping-Bereichsliste vorwärts verkettet.

❑ **prev 0x0004 (4)**

Ein Adreßzeiger auf die *ClipRect*-Datenstruktur des vorangehenden Clipping-Bereichs in der Clipping-Bereichsliste. Über die *prev*Zeiger werden sämtliche Clipping-Bereiche einer Clipping-Bereichsliste rückwärts verkettet.

❑ **lobs 0x0008 (8)**

Ein Adreßzeiger auf die *Layer*-Datenstruktur des Layers, dem dieser Clipping-Bereich zugeordnet ist.

❑ **BitMap 0x000C (12)**

Ein Adreßzeiger auf die *BitMap*-Datenstruktur der Bitmap, in die die Grafikroutinen beim Zugriff auf diesen Clipping-Bereich zeichnen sollen.

❑ **bounds 0x0010 (16)** Eine *Rectangle*-Datenstruktur, deren einzelne Komponenten die Lage und Ausmaße des Clipping-Bereichs innerhalb des übergeordneten Layers angeben.

❑ **_p1 0x0018 (24)** Ein von den Systemroutinen intern für Verwaltungszwecke benötigter Adreßzeiger auf eine *ClipRect*-Datenstruktur.

❑ **_p2 0x001C (28)** Siehe *_p1*.

❑ **reserved 0x0020 (32)** Ein für den Systemgebrauch reserviertes Langwort.

❑ **Flags 0x0024 (36)** Eine 4-Byte-Marke, deren Bit 0 einen für interne Systemzwecke benötigten Status des Clipping-Bereichs anzeigt. Zur Benutzung ist folgende Konstante vordefiniert:

C	Modula-2	Hex	Dez
CR_NEEDS_NO_CONCEALED_RASTERS	NeedsNoConcealedRasters	0001	1

5

CollTable

Offset		C (graphics/gels.h)	Modula-2 (Graphics)
Hex	Dez	struct CollTable {	CollTable = RECORD
0000	0	int (*collPtrs[16])();	collPtrs:ARRAY[0..15] OF AD DRESS
0040	64	};	END;

Assembler Die Datenstruktur ist hier in »graphics/view.i« definiert.
Präfix: *cp_*

Datenstrukturenreferenz »Graphics«: *GelsInfo*.

Routinenreferenz Keine.

Beschreibung

Diese Tabelle dient der Aufnahme der Adressen von sechzehn benutzerdefinierten Routinen zur Kollisionsbehandlung. Diese Routinen finden beim Umgang mit dem Animationssystem mannigfaltige Verwendung.

❑ **collPtrs 0x0000 (0)** Ein Feld von sechzehn Adreßzeigern auf die Codebereiche der Kollisionsroutinen.

ColorMap

Offset		C (graphics/view.h)	Modula-2 (Graphics)
Hex	Dez	struct ColorMap	ColorMap =
		{	RECORD
0000	0	UBYTE Flags;	flags:UByte;
0001	1	UBYTE Type;	type:UByte;
0002	2	UWORD Count;	count:CARDINAL;
0004	4	APTR ColorTable;	colorTable:ADDRESS;
0008	8	};	END;

5

Assembler Präfix: *cm_*

Datenstrukturenreferenz »Graphics«: *IODRPReq*, *ViewPort.*

Routinenreferenz »Graphics«: *FreeColorMap*, *GetColorMap*, *GetRGB4*, *SetRGB4CM.*

Beschreibung

Jeder Viewport besitzt seine eigene Farbpalette, in der die RGB-Zusammensetzung der einzelnen Farben gespeichert ist. Die *ColorMap*-Datenstruktur dient der Verwaltung einer solchen Farbpalette.

❑ **Flags 0x0000 (0)** Eine 1-Byte-Marke, die momentan noch nicht bzw. (intern) nur zwischenzeitlich benutzt wird.

❑ **Type 0x0001 (1)** Eine 1-Byte-Marke, deren Wert den Typ der Farbeinträge dieser Farbpalette festlegt. Bisher ist hierfür lediglich der Wert Null zulässig, wodurch Farbeinträge der Art »xRGB« gekennzeichnet werden. Für die einzelnen Nibble »R«, »G«, »B« (mit Definitionsbereich 0 bis 15) gilt dabei, daß »R« den Rotanteil, »G« den Grünanteil und »B« den Blauanteil der Farbe angibt. Der endgültige Farbwert berechnet sich dann aus $B+16*G+256*R$.

❑ **Count 0x0002 (2)** Eine positive Ganzzahl, deren Wert die Anzahl der gültigen Farbeinträge der Farbpalette angibt.

❑ **ColorTable 0x0004 (4)** Ein Adreßzeiger auf den Anfang einer Tabelle von Farbwerten, die gerade soviel Einträge aufweist, wie der Wert von *Count* angibt.

copinit

Offset		C (graphics/copper.h)	Modula-2 (Graphics)
Hex	Dez	struct copinit {	Copinit = RECORD
0000	0	UWORD diagstrt[4];	diagstrt:ARRAY [0..3] OF CARDINAL;
0008	8	UWORD sprstrtup[40];	sprstrtup:ARRAY [0..39] OF CARDINAL;
0058	88	UWORD sprstop[2];	sprstop:ARRAY [0..1] OF CARDINAL;
005C	92	};	END;

5

Assembler Präfix: *copinit_*

Datenstrukturenreferenz »Graphics«: *GfxBase.*

Routinenreferenz Keine.

Beschreibung

Die *copinit*-Datenstruktur enthält die bereits fertig codierten Datenworte von Copper-Befehlssequenzen, die zur Initialisierung des Displays in Hardware-Copperlisten benötigt werden.

- **diagstrt 0x0000 (0)** Eine codierte Copper-Befehlssequenz für die erste Bitplane.
- **sprstrtup 0x0008 (8)** Eine codierte Copper-Befehlssequenz zur Initialisierung der Sprite-Animation.
- **sprstop 0x0058 (88)** Eine codierte Copper-Befehlssequenz zum Stoppen der Sprite-Animation.

CopIns

Offset		C (graphics/copper.h)	Modula-2 (Graphics)
Hex	Dez	struct CopIns {	CopIns = RECORD
0000	0	short OpCode; union {	CASE opCode:CARDINAL OF | next:
0002	2	struct CopList *nxtlist; struct { union {	nxtlist:CopListPtr; | wait:

Offset		C (graphics/copper.h)	Modula-2 (Graphics)
Hex	Dez		
0002	2	SHORT VWaitPos;	vWaitPos:INTEGER;
0002	2	SHORT DestAddr;	hWaitPos:INTEGER;
		} u1;	
		union	
		{	| move:
0004	4	SHORT HWaitPos;	destAddr:INTEGER;
0004	4	SHORT DestData;	destData:INTEGER;
		} u2;	
		} u4;	
		} u3;	END;
0006	6	};	END;

Assembler Präfix: *ci_*

Datenstrukturenreferenz »Graphics«: *CopList.*

Routinenreferenz Keine.

Beschreibung

Die *CopIns*-Datenstruktur dient sowohl der Aufnahme eines codierten Copper-Befehls als auch dem Verweis auf das nächste Teilstück einer Zwischen-Copperliste.

❑ **OpCode 0x0000 (0)** Die Tag-Marke der Varianten der *CopIns*-Datenstruktur, für die folgende Werte gültig sind:

C	Modula-2	Hex	Dez	Bedeutung
COPPER_MOVE	move	0000	0	Copperbefehl MOVE
COPPER_WAIT	wait	0001	1	Copperbefehl WAIT
CPRNXTBUF	next	0002	2	Verkettungszeiger

❑ **nxtlist 0x0002 (2)** Falls die Tag-Marke *OpCode* gleich *CPRNXTBUF* ist, steht hier ein Adreßzeiger auf den nächsten Abschnitt der Zwischen-Copperliste.

❑ **VWaitPos 0x0000 (2)** Falls die Tag-Marke *OpCode* gleich *COPPER_WAIT* ist, steht hier die vertikale Position des zugehörigen Copper-Befehls *WAIT*.

❑ **DestAddr 0x0002 (2)** Falls die Tag-Marke *OpCode* gleich *COPPER_MOVE* ist, steht hier die Zieladresse des zugehörigen Copper-Befehls *MOVE*.

❑ **HWaitPos 0x0004 (4)**

Falls die Tag-Marke *OpCode* gleich *COPPER_WAIT* ist, steht hier die horizontale Position des zugehörigen Copper-Befehls *WAIT*.

❑ **DestData 0x0004 (4)**

Falls die Tag-Marke *OpCode* gleich *COPPER_MOVE* ist, steht hier das Datenwort des zugehörigen Copper-Befehls *MOVE*.

CopList

Offset Hex	Dez	C (graphics/copper.h)	Modula-2 (Graphics)
		struct CopList	CopList =
		{	RECORD
0000	0	struct CopList *Next;	next:CopListPtr;
0004	4	struct CopList *_CopList;	copList:CopListPtr;
0008	8	struct ViewPort *_ViewPort;	viewPort:ViewPortPtr;
000C	12	struct CopIns *CopIns;	copIns:CopInsPtr;
0010	16	struct CopIns *CopPtr;	copPtr:CopInsPtr;
0014	20	UWORD *CopLStart;	copLStart:ADDRESS;
0018	24	UWORD *CopSStart;	copSStart:ADDRESS;
001C	28	SHORT Count;	count:INTEGER;
001E	30	SHORT MaxCount;	maxCount:INTEGER;
0020	32	SHORT DyOffset;	dyOffset:INTEGER;
0022	34	};	END;

Assembler Präfix: *cl_*

Datenstrukturenreferenz »Graphics«: *CopIns, CopList, UCopList, ViewPort.*

Routinenreferenz »Graphics«: *FreeCopList.*

Beschreibung

Die *CopList*-Datenstruktur dient zur Verwaltung der Zwischen-Copperlisten, aus denen die »Graphics«-Systemroutine *MrgCop* die Hardware-Copperlisten berechnet, die zur Erzeugung des Monitorausgabebildes benötigt werden. Ein Viewport besitzt mehrere Zwischen-Copperlisten und auch noch eine Benutzer-Copperliste. Alle diese Copperlisten enthalten natürlich eigene Copper-Befehlssequenzen, deren Reihenfolge noch nicht aufeinander abgestimmt ist. Innerhalb der Hardware-Copperlisten eines Views muß dagegen die Reihenfolge stimmen, weil sonst das Ausgabebild nicht korrekt erzeugt werden kann. Daher werden die diversen Copperlisten eines Viewports in Form von Zwischen-Copperlisten verwaltet, weil aus diesen die endgültige Hardware-Copperliste des Views schnell und einfach in der richtigen Reihenfolge zusammengebaut werden kann.

❑ **Next 0x0000 (0)**

Ein Adreßzeiger auf die nächste *CopList*-Datenstruktur der Zwischen-Copperliste, die einen neuen Abschnitt repräsentiert. Über die *Next*-Zeiger werden sämtliche Abschnitte einer Zwischen-Copperliste vorwärts verkettet.

❑ **_CopList 0x0004 (4)** Ein vom System während des Zusammenbaus der Hardware-Copperliste aus den diversen Zwischen-Copperlisten intern benötigter Adreßzeiger auf eine *CopList*-Datenstruktur.

❑ **_ViewPort 0x0008 (8)** Ein vom System während des Zusammenbaus der Hardware-Copperliste aus den diversen Zwischen-Copperlisten intern benötigter Adreßzeiger auf eine *ViewPort*-Datenstruktur.

❑ **CopIns 0x000C (12)** Ein Adreßzeiger auf den ersten *CopIns*-Eintrag in der Copper-Befehlsliste dieser Zwischen-Copperliste.

❑ **CopPtr 0x0010 (16)** Ein Adreßzeiger auf den gerade bearbeiteten *CopIns*-Eintrag der Copper-Befehlsliste dieser Zwischen-Copperliste.

❑ **CopLStart 0x0014 (20)** Ein von der »Graphics«-Systemroutine *MrgCop* einzutragender Adreßzeiger auf die codierte Copper-Befehlssequenz für den »long frame« (siehe ViewPort).

❑ **CopSStart 0x0018 (24)** Ein von der »Graphics«-Systemroutine *MrgCop* einzutragender Adreßzeiger auf die codierte Copper-Befehlssequenz für den »short frame« (siehe ViewPort).

❑ **Count 0x001C (28)** Eine positive Ganzzahl, deren Wert die Anzahl der bisher in diese Zwischen-Copperliste eingetragenen Copper-Befehle angibt.

❑ **MaxCount 0x001E (30)** Eine positive Ganzzahl, deren Wert die maximale Anzahl von Copper-Befehlen festlegt, die in dieser Zwischen-Copperliste Platz finden.

❑ **DyOffset 0x0020 (32)** Eine Ganzzahl, deren Wert den vertikalen Offset der in dieser Zwischen-Copperliste enthaltenen *WAIT*-Befehle angibt.

cprlist

Offset		C (graphics/copper.h)	Modula-2 (Graphics)
Hex	**Dez**	struct cprlist	Cprlist =
		{	RECORD
0000	0	struct cprlist *Next;	next:CprlistPtr;
0004	4	UWORD *start;	start:ADDRESS;
0008	8	SHORT MaxCount;	maxCount:INTEGER;
000A	10	};	END;

Assembler Präfix: *crl_*

Datenstrukturenreferenz »Graphics«: *CprList*, *View*.

Routinenreferenz »Graphics«: *FreeCprlist.*

Beschreibung

Mit Hilfe der *cprlist*-Datenstruktur werden von der Systemsoftware die Hardware-Copperlisten verwaltet.

- **Next 0x0000 (0)** — Ein Adreßzeiger auf die nächste *cprlist*-Datenstruktur in dieser Liste von Hardware-Copperlisten. Über die *Next*-Zeiger werden sämtliche (Einzel-)Hardware-Copperlisten einer *cprlist*-Datenstruktur vorwärts verkettet.
- **start 0x0004 (4)** — Ein Adreßzeiger auf den Anfang einer tatsächlichen Hardware-Copperliste, in der die codierte Copper-Befehlssequenz zu finden ist.
- **MaxCount 0x0008 (8)** — Eine positive Ganzzahl, deren Wert der Anzahl von Copper-Befehlen (das sind jeweils Langworte) in der zugehörigen Hardware-Copperliste entspricht.

5

DBufPacket

Offset Hex	Offset Dez	C (graphics/gels.h)	Modula-2 (Graphics)
		struct DBufPacket	DBufPacket =
		{	RECORD
0000	0	WORD BufY;	bufY:INTEGER;
0002	2	WORD BufX;	bufX:INTEGER;
0004	4	struct VSprite *BufPath;	bufPath:VSpritePtr;
0008	8	WORD *BufBuffer;	bufBuffer:ADDRESS;
000C	12	};	END;

Assembler Präfix: *dbp_*
Syntax: Strukturname = *DBP*

Datenstrukturenreferenz »Graphics«: *Bob.*

Routinenreferenz Keine.

Beschreibung

Bei der Animation von Bobs besteht die Möglichkeit, diese von den Animationsroutinen mit doppelter Pufferung bewegen zu lassen. Dabei nimmt die *DBufPacket*-Datenstruktur dann die für das Bob neben der *Bob*-Datenstruktur noch zusätzlich nötigen Informationen auf. Die spezielle *DBufPacket*-Datenstruktur eines Bobs wird mit diesem über den Adreßzeiger *DBuffer* seiner *Bob*-Datenstruktur verknüpft.

5

❑ **BufY 0x0000 (0)**
Eine Ganzzahl, deren Wert die y-Koordinate der letzten Position des Bobs angibt, an der der Hintergrund im normalen *Savebuffer* des Bobs (siehe *Bob*-Datenstruktur) gespeichert wurde.

❑ **BufX 0x0002 (2)**
Eine Ganzzahl, deren Wert die x-Koordinate der letzten Position des Bobs angibt, an der der Hintergrund im normalen *Savebuffer* des Bobs gespeichert wurde.

❑ **BufPath 0x0004 (4)**
Ein Adreßzeiger auf eine *VSprite*-Datenstruktur, der von den Animationsroutinen zur korrekten Einhaltung der Darstellungsreihenfolge aller Bobs der GEL-Liste benötigt wird. Der Wert dieses Zeigers hängt eng mit den Adreßzeigern *ClearPath* und *DrawPath* aus der *VSprite*-Datenstruktur des Bobs zusammen.

❑ **BufBuffer 0x0008 (8)**
Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der mindestens so groß sein muß wie der normale *Savebuffer* des Bobs (siehe *Bob*-Datenstruktur). Dieser Speicherbereich dient wie der *SaveBuffer* zur Aufnahme der Playfield-Hintergrunddaten, die bei der Darstellung eines Bobs überschrieben werden. Im doppelt gepufferten Darstellungsmodus werden also zwei verschiedene Pufferspeicher zur Aufnahme der Hintergrunddaten verwendet.

GelsInfo

Offset		**C (graphics/rastport.h)**	**Modula-2 (Graphics)**
Hex	**Dez**	struct GelsInfo	GelsInfo =
		{	RECORD
0000	0	BYTE sprRsrvd;	sprRsrvd:Byte;
0001	1	UBYTE Flags;	flags:Byte;
0002	2	struct VSprite *gelHead;	gelHead:VSpritePtr;
0006	6	struct VSprite *gelTail;	gelTail:VSpritePtr;
000A	10	WORD *nextLine;	nextLine:ADDRESS;
000E	14	WORD **lastColor;	lastColor:ADDRESS;
0012	18	struct collTable *collHandler;	collHandler:CollTablePtr;
0016	22	short leftmost;	leftmost:INTEGER;
0018	24	short rightmost;	rightmost:INTEGER;
001A	26	short topmost;	topmost:INTEGER;
001C	28	short bottommost;	bottommost:INTEGER;
001E	30	APTR firstBlissObj;	firstBlissObj:ADDRESS;
0022	34	APTR lastBlissObj;	lastBlissObj:ADDRESS;
0026	38	};	END;

Assembler	Präfix:	*gi_*
Datenstrukturenreferenz	»Graphics«:	*RastPort.*
Routinenreferenz	»Graphics«:	*InitGels, SetCollision.*

Beschreibung

Mit Hilfe der *GelsInfo*-Datenstruktur verwaltet das Animationssystem eine Liste von GELs = <g>raphic <el>ements, die zusammen im gleichen Displaybereich dargestellt werden sollen. Der gemeinsame Displaybereich für GELs muß auf jeden Fall ein Rastport sein! Die spezielle *GelsInfo*-Datenstruktur eines Rastports wird mit diesem über den Adreßzeiger *GelsInfo* seiner *RastPort*-Datenstruktur verknüpft. Aus den in der *GelsInfo*-Datenstruktur enthaltenen Informationen berechnet insbesondere die Spritemaschine die tatsächlichen Zuordnungen der aktivierten virtuellen Sprites.

- **sprRsrvd 0x0000 (0)** Eine 1-Byte-Marke, deren einzelne Bits festlegen, welche der Hardwaresprites 0 bis 7 von der Spritemaschine für die Zuweisung an virtuelle Sprites nicht benutzt werden dürfen.

- **Flags 0x0001 (1)** Eine intern für den Systemgebrauch der Animationsroutinen bestimmte 1-Byte-Marke.

- **gelHead 0x0002 (2)** Ein Adreßzeiger auf die *VSprite*-Datenstruktur des ersten Elements der GEL-Liste dieses *Gelsinfo*s. Beachten Sie bitte, daß dieses Anfangselement lediglich als Dummy-GEL fungiert und nicht gezeichnet wird.

- **gelTail 0x0006 (6)** Ein Adreßzeiger auf die *VSprite*-Datenstruktur des letzten Elements der GEL-Liste dieses *Gelsinfo*s. Beachten Sie bitte, daß dieses Endelement lediglich als Dummy-GEL fungiert und nicht gezeichnet wird.

- **nextLine 0x000A (10)** Ein Adreßzeiger auf ein Feld von acht Ganzzahlen, deren Werte die Zeilennummern angeben, bei denen das entsprechende Hardwaresprite wieder für eine Neuzuordnung verfügbar ist. Dabei beschreibt das Feldelement 0 die nächste freie Zeile für Hardwaresprite 0 usw. bis hin zu Feldelement 7, das für Hardwaresprite 7 zuständig ist. Diese Angaben bzgl. der nächsten freien Zeilen benötigt die Spritemaschine zur flexiblen und gleichzeitig korrekten Zuordnung aller virtuellen Sprites dieser GEL-Liste.

- **lastColor 0x000E (14)** Ein Adreßzeiger auf ein Feld von weiteren acht Adreßzeigern, die auf die für die entsprechenden Hardwaresprites zuletzt benutzten (Mini-)Farbtabellen verweisen. Für die Zuordnung der Feldelemente zu den einzelnen Hardwaresprites siehe Erläuterungen zum Adreßzeiger *nextLine*.

- **collHandler 0x0012 (18)** Ein Adreßzeiger auf eine *collTable*-Datenstruktur, in der die Adressen der sechzehn benutzerdefinierten Kollisionsroutinen gespeichert sind, die für Kollisionsbehandlung sämtlicher GELs dieser GEL-Liste zuständig sind.

- **leftmost 0x0016 (22)** Eine Ganzzahl, deren Wert die x-Koordinate des linken Rands des Clipping-Bereichs dieser GEL-Liste festlegt. Dieser Wert wird von den Animationsroutinen benötigt, um die Kollision eines GELs mit den Randbegrenzungen des Clipping-Bereichs feststellen zu können.

- **rightmost 0x0018 (24)** Eine Ganzzahl, deren Wert die x-Koordinate des rechten Rands des Clipping-Bereichs dieser GEL-Liste festlegt. Dieser Wert wird von den Animationsroutinen benötigt, um die Kollision eines GELs mit den Randbegrenzungen des Clipping-Bereichs feststellen zu können.

- **topmost 0x001A (26)** Eine Ganzzahl, deren Wert die y-Koordinate des oberen Rands des Clipping-Bereichs dieser GEL-Liste festlegt. Dieser Wert wird von den Animationsroutinen benötigt, um die Kollision eines GELs mit den Randbegrenzungen des Clipping-Bereichs feststellen zu können.

- **bottommost 0x001C (28)** Eine Ganzzahl, deren Wert die y-Koordinate des unteren Rands des Clipping-Bereichs dieser GEL-Liste festlegt. Dieser Wert wird von den Animationsroutinen benötigt, um die Kollision eines GELs mit den Randbegrenzungen des Clipping-Bereichs feststellen zu können.

- **firstBlissObj 0x001E (30)** Ein für den Gebrauch durch die Animationsroutinen bestimmter Adreßzeiger, der normalerweise auf das erste Bob dieser GEL-Liste verweist.

- **lastBlissObj 0x0022 (34)** Ein für den Gebrauch durch die Animationsroutinen bestimmter Adreßzeiger, der normalerweise auf das letzte Bob dieser GEL-Liste verweist.

GfxBase

Offset		C (graphics/gfxbase.h)	Modula-2 (Graphics)
Hex	Dez	struct GfxBase	GfxBase =
		{	RECORD
0000	0	struct Library LibNode;	libNode:Library;
0022	34	struct View *ActiView;	actiView:ViewPtr;
0026	38	struct copinit *copinit;	copinit:CopinitPtr;
002A	42	long *cia;	cia:ADDRESS;
002E	46	long *blitter;	blitter:ADDRESS;
0032	50	UWORD *LOFlist;	loFlist:ADDRESS;
0036	54	UWORD *SHFlist;	shFlist:ADDRESS;
003A	58	struct bltnode *blthd;	blthd:BltnodePtr;
003E	62	struct bltnode *bltl;	bltl:BltnodePtr;

Offset		C (graphics/gfxbase.h)	Modula-2 (Graphics)
Hex	Dez		
0042	66	struct bltnode *bsblthd;	bsblthd:BltnodePtr;
0046	70	struct bltnode *bsbltl;	bsbltl:BltnodePtr;
004A	74	struct Interrupt vbsrv;	vbsrv:Interrupt;
0060	96	struct Interrupt timsrv;	timsrv:Interrupt;
0076	118	struct Interrupt bltsrv;	bltsrv:Interrupt;
008C	140	struct List TextFonts;	textFonts:List;
009A	154	struct TextFont *DefaultFont;	defaultFont:TextFontPtr;
009E	158	UWORD Modes;	modes:BITSET;
00A0	160	BYTE VBlank;	vBlank:UByte;
00A1	161	BYTE Debug;	debug:BYTE;
00A2	162	SHORT BeamSync;	beamSync:INTEGER;
00A4	164	SHORT system_bplcon0;	bplcon0:BITSET;
00A6	166	UBYTE SpriteReserved;	spriteReserved:UByte;
00A7	167	UBYTE bytereserved;	bytereserved:UByte;
00A8	168	USHORT Flags;	flags:BITSET;
00AA	170	SHORT BlitLock;	blitLock:INTEGER;
00AC	172	short BlitNest;	blitNest:INTEGER;
00AE	174	struct List BlitWaitQ;	blitWaitQ:List;
00BC	188	struct Task *BlitOwner;	blitOwner:TaskPtr;
00C0	192	struct List TOF_WaitQ;	waitQ:List;
00CE	206	UWORD DisplayFlags;	displayFlags:DisplayFlagSet;
00D0	208	struct SimpleSprite **SimpleSprites;	simpleSprites:ADDRESS;
00D4	212	UWORD MaxDisplayRow;	maxDisplayRow:CARDINAL;
00D6	214	UWORD MaxDisplayColumn;	maxDisplayColumn:CARDINAL;
00D8	216	UWORD NormalDisplayRows;	normalDisplayRows:CARDINAL;
00DA	218	UWORD NormalDisplayColumns;	normalDisplayColumns:CARDINAL;
00DC	220	UWORD NormalDPMX;	normalDPMX:CARDINAL;
00DE	222	UWORD NormalDPMY;	normalDPMY:CARDINAL;
00E0	224	struct SignalSemaphore *LastChanceMemory;	lastChanceMemory: SignalSemaphorePtr;
00E4	228	UWORD *LCMptr;	lcMptr:ADDRESS;
00E8	232	UWORD MicrosPerLine;	microsPerLine:CARDINAL;
00EA	234	UWORD MinDisplayColumn;	reserved:
00EC	236	ULONG reserved[23];	ARRAY [0..1] OF LONGCARD;
00F2	328	};	END;

Assembler — Präfix: *gb_*
Syntax: Komponente *LibNode* ist nicht definiert!

Datenstrukturenreferenz — »Intuition«: *IntuitionBase*.

Routinenreferenz — »Exec«: *OldOpenLibrary*, *OpenLibrary*.

Beschreibung

Die *GfxBase*-Datenstruktur stellt die Basisdatenstruktur der »graphics.library« dar, die beim ersten Öffnen dieser Bibliothek erzeugt wird. In der *GfxBase*-Datenstruktur sind der momentan aktive View, die Display-Copperlisten, die Systemfontliste, die diversen Job-Listen des Blitters, die drei (Standard-) Interrupt-Server, die den Hardwaresprites zugeordneten *SimpleSprite*-Datenstrukturen, eine Liste auf den Blitter wartender Tasks sowie eine Liste auf das »vertical blanking« Intervall wartender Tasks verankert. Darüber hinaus finden sich hier eine Vielzahl weiterer Statusinformationen bezüglich der verfügbaren Displayhardware und des aktuellen Displays.

- **LibNode 0x0000 (0)** — Eine *Library*-Datenstruktur (siehe »Exec«), die den Eintrag der »graphics.library« in der globalen (Library-)Liste der Systembibliotheken darstellt.

- **ActiView 0x0022 (34)** — Ein Adreßzeiger auf die *View*-Datenstruktur des gerade geladenen und damit sichtbaren Views.

- **copinit 0x0026 (38)** — Ein Adreßzeiger auf die *copinit*-Datenstruktur, deren Hardware-Copperlisten momentan für den Copper-Startup verwendet werden.

- **cia 0x002A (42)** — Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der als Resource für den CIA-Baustein 8520 dient.

- **blitter 0x002E (46)** — Ein bisher nicht benutzter Adreßzeiger auf einen Speicherbereich im Chip-RAM, der dem Blitter zukünftig als Resource dienen soll.

- **LOFlist 0x0032 (50)** — Ein Adreßzeiger auf die codierten Daten der Hardware-Copperliste zur Darstellung des »long frame« (siehe Viewport).

- **SHFlist 0x0036 (54)** — Ein Adreßzeiger auf die codierten Daten der Hardware-Copperliste zur Darstellung des »short frame« (siehe Viewport).

- **blthd 0x003A (58)** — Ein Adreßzeiger auf die *bltnode*-Datenstruktur des Anfangselements der FIFO-Schlange von unerledigten Blitterjobs, die durch Aufrufe der »Graphics«-Routine *QBlit* in Auftrag gegeben wurden.

- **blttl 0x003E (62)** — Ein Adreßzeiger auf die *bltnode*-Datenstruktur des letzten Elements der FIFO-Schlange von unerledigten Blitterjobs, die durch Aufrufe der »Graphics«-Routine *QBlit* in Auftrag gegeben wurden.

- **bsblthd 0x0042 (66)** — Ein Adreßzeiger auf die *bltnode*-Datenstruktur des Anfangselements der FIFO-Schlange von unerledigten BS-Blitterjobs, die durch Aufrufe der »Graphics«-Routine *QBSBlit* in Auftrag gegeben wurden. Die Abkürzung »QBS« steht dabei für »queue beam synchronized«.

- **bsblttl 0x0046 (70)** — Ein Adreßzeiger auf die *bltnode*-Datenstruktur des letzten Elements der FIFO-Schlange von unerledigten BS-Blitter-

jobs, die durch Aufrufe der »Graphics«-Routine *QBSBlit* in Auftrag gegeben wurden. Die Abkürzung »QBS« steht dabei für »queue beam synchronized«.

- **vbsrv 0x004A (74)** Eine *Interrupt*-Datenstruktur (siehe »Exec«), die den für »vertical blanking«-Unterbrechungen zuständigen Interrupt-Server identifiziert.
- **timsrv 0x0060 (96)** Eine *Interrupt*-Datenstruktur (siehe »Exec«), die den für Timer-Unterbrechungen zuständigen Interrupt-Server identifiziert.
- **bltsrv 0x0076 (118)** Eine *Interrupt*-Datenstruktur (siehe »Exec«), die den für Blitter-Unterbrechungen zuständigen *Interrupt*-Server identifiziert.
- **TextFonts 0x008C (140)** Eine *List*-Datenstruktur (siehe »Exec«), die der Verwaltung der Systemfontliste dient. In diese Liste werden alle momentan verfügbaren Textfonts über die Komponente *mn_Node* der *tf_Message* ihrer *TextFont*-Datenstruktur eingereiht.
- **DefaultFont 0x009A (154)** Ein Adreßzeiger auf die *TextFont*-Datenstruktur des Textfonts, der von den »Graphics«-Routinen standardmäßig verwendet wird.
- **Modes 0x009E (158)** Laut [RKA] findet sich hier eine Kopie des aktuellen Werts des *BPLCON0*-Registers (siehe Datenstruktur *Custom* in »Hardware«).
- **VBlank 0x00A0 (160)** ??? (Hat offensichtlich etwas mit dem »vertical blanking« Intervall zu tun!)
- **Debug 0x00A1 (161)** ??? (Habe meinen Wellensittich befragt... Er meint das geht in Richtung »Debugger Wack«.)
- **BeamSync 0x00A2 (162)** ??? (Vermutung: Entsprechender *beamsync*-Wert des nächsten *blitnode*-Blitterjobs.)
- **system_bplcon0 0x00A4 (164)** Laut [RKA] wird dieser Wert für jedes Display ins *BPLCON0*-Register geODERt.
- **SpriteReserved 0x00A6 (166)** Eine 1-Byte Marke, deren einzelne Bits angeben, welche der Hardwaresprites 0 bis 7 momentan für den exklusiven Gebrauch durch irgendwelche Tasks reserviert sind.
- **bytereserved 0x00A7 (167)** Eine für den Systemgebrauch reservierte 1-Byte-Marke.
- **Flags 0x00A8 (168)** ??? (Unser Vorschlag an Commodore: Ein wenig Glasnost könnte hier nicht schaden!)
- **BlitLock 0x00AA (170)** Eine Ganzzahl, deren Wert als Sperrmarkierung bei der exklusiven Benutzung des Blitters durch einen Task fungiert.
- **BlitNest 0x00AC (172)** Eine Ganzzahl, die als Zähler für verschachtelte Anforderungen exklusiver Blitter-Benutzung benötigt wird.

5

❑ **BlitWaitQ 0x00AE (174)**

Eine *List*-Datenstruktur (siehe »Exec»), die der Verwaltung der Liste von Tasks dient, die momentan auf die Benutzung des Blitters warten.

❑ **BlitOwner 0x00BC (188)**

Ein Adreßzeiger auf die *Task*-Datenstruktur (siehe »Exec«) der Task, die den Blitter gerade exklusiv für sich beansprucht. Angemeldet werden entsprechende Blitter-Anforderungen durch einen Aufruf der »Graphics«-Routine *OwnBlit*.

❑ **TOF_WaitQ 0x00C0 (192)**

Eine *List*-Datenstruktur (siehe »Exec«), die der Verwaltung der Liste von Tasks dient, die im Moment auf das nächste »vertical blanking«-Intervall warten. In diese Liste werden die Tasks bei einem Aufruf der »Graphics«-Routine *WaitTOF* eingereiht. Dabei steht die Abkürzung »TOF« für »top of frame«.

❑ **DisplayFlags 0x00CE (206)**

Eine 2-Byte-Marke, deren einzelne Bits Informationen über die angeschlossene Display-Hardware enthalten. Diese Bits werden beim Einschalten des Amigas bestimmt. Für die Benutzung sind dabei folgende Konstanten vordefiniert:

C	Modula-2	Hex	Dez	Bedeutung
NTSC	ntsc	0001	1	Ist ein NTSC-Monitor angeschlossen?
GENLOC	genloc	0002	2	Ist ein GENLOC-Interface verfügbar?
PAL	pal	0004	4	Ist ein PAL-Monitor angeschlossen?

Wichtig: In Modula-2 ist hier der Mengentyp *DisplayFlagSet* zu benutzen.

❑ **SimpleSprites 0x00D0 (208)**

Ein Adreßzeiger auf ein Feld von acht Adreßzeigern, die auf die *SimpleSprite*-Datenstrukturen verweisen, die den Hardwaresprites 0 bis 7 momentan zugeordnet sind.

❑ **MaxDisplayRow 0x00D4 (212)**

Eine positive Ganzzahl, deren Wert die Höhe des größtmöglichen Low-Resolution-Non-Interlace-Displays inkl. Overscan-Zeilen angibt.

❑ **MaxDisplayColumn 0x00D6 (214)**

Eine positive Ganzzahl, deren Wert die Breite des größtmöglichen Low-Resolution-Non-Interlace-Displays inkl. Overscan-Zeilen angibt.

❑ **NormalDisplayRows 0x00D8 (216)**

Eine positive Ganzzahl, deren Wert die Höhe des aktuellen Standarddisplays in Pixeln festlegt.

❑ **NormalDisplayColumns 0x00DA(216)**

Eine positive Ganzzahl, deren Wert die Breite des aktuellen Standarddisplays in Pixeln festlegt.

- ❑ **NormalDPMX 0x00DC (220)** Eine positive Ganzzahl, deren Wert die Breite eines Pixels für ein Low-Resolution-Non-Interlace-Display definiert, indem er angibt, wie viele Pixel pro Meter benötigt werden. Die Abkürzung »DPM« bedeutet »dots per meter«.
- ❑ **NormalDPMY 0x00DE (222)** Eine positive Ganzzahl, deren Wert die Höhe eines Pixels für ein Low-Resolution-Non-Interlace-Display definiert, indem er angibt, wie viele Pixel pro Meter benötigt werden. Die Abkürzung »DPM« bedeutet »dots per meter«.
- ❑ **LastChanceMemory 0x00E0 (224)** Eine *SignalSemaphore*-Datenstruktur (siehe »Exec«), die den Zugriff mehrerer Tasks auf das »last chance memory« synchronisiert.
- ❑ **LCMptr 0x00E4 (228)** ??? (Bedeutet die Abkürzung »LCM« etwa »last chance memory«, dann könnte dies ein Adreßzeiger auf einen Speicherbereich für den äußersten Notfall sein.)
- ❑ **MicrosPerLine 0x00E8 (232)** Eine positive Ganzzahl, deren Wert die Anzahl von Mikrosekunden angibt, die zum Aufbau eines Videobilds benötigt werden.
- ❑ **MinDisplayColumn 0x00EA (234)** ??? (Wen interessieren solche Displaydetails?)
- ❑ **reserved 0x00EC (236)** Laut [RKA] sind diese Datenbytes für zukünftige Erweiterungen reserviert.

5

Isrvstr

Offset		C (graphics/graphint.h)	Modula-2 (Graphics)
Hex	**Dez**	struct Isrvstr	Isrvstr =
		{	RECORD
0000	0	struct Node is_Node;	node:Node;
000E	14	struct Isrvstr *Iptr;	iptr:IsrvstrPtr;
0012	18	int (*code)();	code:ADDRESS;
0016	22	int (*ccode)();	ccode:ADDRESS;
001A	26	int Carg;	carg:INTEGER;
001C	28	};	END;

Assembler Diese systeminterne Datenstruktur ist in Assembler nicht definiert.

Datenstrukturenreferenz »Graphics«: *Isrvstr.*

Routinenreferenz Keine.

Beschreibung

Wird von privater »Graphics«-Systemroutine *AddTOFTask* dazu benutzt, um Tasks nach Aufruf von *WaitTOF* an den Vertical-Blank-Interrupt-Server anzuhängen!

- **is_Node 0x0000 (0)** Eine eingebundene *Node*-Datenstruktur zur Verknüpfung mehrerer solcher Datenstrukturen.
- **Iptr 0x000E (14)** Ein Adreßzeiger auf weitere *Isrvstr*-Datenstrukturen.
- **code 0x0012 (18)** Ein Adreßzeiger auf eine systeminterne Routine.
- **ccode 0x0016 (22)** Ein Adreßzeiger auf eine weitere systeminterne Routine.
- **Carg 0x001A (26)** ???

Layer

Offset		C (graphics/clip.h)	Modula-2 (Graphics)
Hex	**Dez**	struct Layer	Layer =
		{	RECORD
0000	0	struct Layer *front;	front:LayerPtr;
0004	4	struct Layer *back;	back:LayerPtr;
0008	8	struct ClipRect *ClipRect;	clipRect:ClipRectPtr;
000C	12	struct RastPort *rp;	rp:RastPortPtr;
0010	16	struct Rectangle bounds;	bounds:Rectangle;
0018	24	UBYTE reserved[4];	reserved:ARRAY [0..3] OF BYTE;
001C	28	UWORD priority;	priority:CARDINAL;
001E	30	UWORD Flags;	flags:CARDINAL;
0020	32	struct BitMap *SuperBitMap;	superBitMap:BitMapPtr;
0022	34	struct ClipRect *SuperClipRect;	superClipRect:ClipRectPtr;
0026	38	APTR Window;	window:ADDRESS;
002A	42	SHORT Scroll_X;	scrollX:INTEGER;
002C	44	SHORT Scroll_Y;	scrollY:INTEGER;
0030	48	struct ClipRect *cr;	cr:ClipRectPtr;
0034	52	struct ClipRect *cr2;	cr2:ClipRectPtr;
0038	56	struct ClipRect *crnew;	crnew:ClipRectPtr;
003C	60	struct ClipRect *SuperSaveClipRects;	superSaveClipRects:ClipRectPtr;
0040	64	struct ClipRect *_cliprects;	cliprects:ClipRectPtr;
0044	68	struct Layer_Info *LayerInfo;	layerInfo:LayerInfoPtr;
0048	72	struct SignalSemaphore Lock;	lock:SignalSemaphore;
0076	118	UBYTE reserved3[8];	reserved3:ARRAY [0..7] OF BYTE;
007E	126	struct Region *ClipRegion;	clipRegion:RegionPtr;
0082	130	struct Region *saveClipRects;	saveClipRects:RegionPtr;
0086	134	UBYTE reserved2[22];	reserved2:ARRAY [0..21] OF BYTE;
009C	156	struct Region *DamageList;	damageList:RegionPtr;
00A0	160	};	END;

Assembler

Präfix: *lr_*
Syntax: *struct Rectangle bounds —> UWORD lr_MinX, lr_MinY, lr_MaxX, lr_MaxY*

Datenstrukturenreferenz

»Graphics«: *Layer, LayerInfo, RastPort.*
»Intuition«: *GadgetInfo, GListEnv, Requester, Screen, StringInfo, Window.*

Routinenreferenz

»Graphics«: *AttemptLockLayerRom, CopySBitMap, LockLayerRom, SyncSBitMap, Unlock-LayerRom.*

»Layers«: *BeginUpdate, BehindLayer, Create-BehindLayer, CreateUpfrontLayer, DeleteLayer, EndUpdate, InstallClip-Region, Locklayer, MoveLayer, Move-LayerInFrontOf, ScrollLayer, Size-Layer, UnlockLayer, UpfrontLayer, WhichLayer.*

Beschreibung

Die *Layer*-Datenstruktur dient dazu, innerhalb eines Viewports bzw. einer Bitmap einen rechteckigen Arbeitsbereich festzulegen. Diese Arbeitsbereiche sind insofern stapelbar, als daß sie sich mit anderen überlappen dürfen. Dabei sind insbesondere beliebige Überlappungen sowohl in horizontaler als auch in vertikaler Richtung erlaubt. Der Zweck der Layerbildung liegt darin, daß die Grafikroutinen mit ihnen in dem Sinn zusammenarbeiten, daß sie auch in verdeckte Teile des Layers zeichnen. Daher stellen die Layer zusammen mit der »layers.library« die eigentliche Basis des Fenstersystems von Intuition dar.

Um eine korrekte Behandlung der Überlappungsbereiche zu gewährleisten, besteht ein Layer vornehmlich aus einer Menge (Liste!) von Clipping-Bereichen. Diese enthalten in Form von *ClipRect*-Datenstrukturen die Informationen darüber, in welche Speicherbereiche die Grafikroutinen zeichnen sollen. Weiterhin werden innerhalb eines Layers noch Damagelisten verwaltet. Dort sind sämtliche Bereiche des Layers eingetragen, die zwischenzeitlich einmal verdeckt waren und daher eventuell einer Auffrischung bedürfen.

❑ **front 0x0000 (0)**

Ein Adreßzeiger auf die *Layer*-Datenstruktur des Layers, der momentan unmittelbar über diesem Layer liegt. Über die *front*-Zeiger werden sämtliche Layer eines *Layerinfo*s in der Reihenfolge ihrer augenblicklichen Lage vorwärts verkettet.

❑ **back 0x0004 (4)**

Ein Adreßzeiger auf die *Layer*-Datenstruktur des Layers, der momentan unmittelbar unter diesem Layer liegt. Über die *back*-Zeiger werden sämtliche Layer eines *Layerinfo*s in der Reihenfolge ihrer augenblicklichen Lage rückwärts verkettet.

❑ **ClipRect 0x0008 (8)**

Ein Adreßzeiger auf die *ClipRect*-Datenstruktur des ersten Clipping-Bereichs in der Clipping-Bereichsliste dieses Layers.

❑ **rp 0x000C (12)**

Ein Adreßzeiger auf die *RastPort*-Datenstruktur des zu diesem Layer gehörigen Rastports.

❑ **bounds 0x0010 (16)** Eine *Rectangle*-Datenstruktur, deren Komponenten die Ausmaße und Lage dieses Layers innerhalb des Viewports bzw. der Bitmap angeben.

❑ **reserved 0x0018 (24)** Ein für den Systemgebrauch reserviertes Feld von vier Bytes.

❑ **priority 0x001C (28)** Eine positive Ganzzahl, deren Wert eine von den Systemroutinen intern benötigte Prioritätskennung des Layers festlegt.

❑ **Flags 0x001E (30)** Eine 2-Byte-Marke, deren einzelne Bits gemäß der folgenden Aufstellung Auskunft über den Zustand und die Art des Layers geben.

5

Hex	Dez	C	Modula-2	Bedeutung
0001	1	LAYERSIMPLE	layerSimple	Simple-Refresh Layer
0002	2	LAYERSMART	layerSmart	Smart-Refresh Layer
0004	4	LAYERSUPER	layerSuper	Superbitmap-Layer
0008	8	LAYERUPDATING	layerUpdating	Ist gerade ein Update in Gang?
0010	16	LAYERBACKDROP	layerBackdrop	Backdrop-Layer
0040	64	LAYERREFRESH	layerRefresh	Läuft gerade ein Refresh?
0100	256	LAYER_CLIPRECTS_LOST	layerClipRectsLost	Verlust von Clipping-Bereichen durch Speicherprobleme bei vormaliger Layer-Operation

Wichtig: In Modula-2 ist hier der Mengentyp *LayerFlagSet* zu benutzen.

❑ **SuperBitMap 0x0020 (32)** Im Falle eines Superbitmap-Layers steht hier ein Adreßzeiger auf die *BitMap*-Datenstruktur der benötigten Zusatzbitmap.

❑ **SuperClipRect 0x0022 (34)** Ein Adreßzeiger auf eine *ClipRect*-Datenstruktur, der im Falle eines Superbitmap-Layers auf den ersten Clipping-Bereich in der Clipping-Bereichsliste der Superbitmap verweist. Anderenfalls ist er als Zeiger auf das Anfangselement der Damage-Clippingliste zu deuten, die beim automatischen Refresh vom System benötigt wird.

❑ **Window 0x0026 (38)** Ein Adreßzeiger auf die Datenstruktur, die dem zu diesem Layer gehörigen Fenster zugeordnet ist. Bei der Arbeit mit der Intuition-Oberfläche finden Sie hier demnach einen Adreßzeiger auf die *Window*-Datenstruktur des Intuition-Fensters.

❑ **Scroll_X 0x002A (42)** Eine Ganzzahl, deren Wert dem horizontalen Verschiebesatz in Pixeln entspricht, der diesem Layer mittels des letzten Aufrufs von *ScrollLayer* zugeordnet wurde.

- **Scroll_Y 0x002C (44)** Eine Ganzzahl, deren Wert dem vertikalen Verschiebesatz in Pixeln entspricht, der diesem Layer mittels des letzten Aufrufs von *ScrollLayer* zugeordnet wurde.
- **cr 0x0030 (48)** Ein Adreßzeiger auf eine *ClipRect*-Datenstruktur, der von der privaten Systemroutine dedice zur Verwaltung und Analyse der diversen Überlappungsbereiche verwendet wird.
- **cr2 0x0034 (52)** Wie Komponente *cr*.
- **crnew 0x0038 (56)** Wie Komponente *cr*, allerdings dient dieser Zeiger vornehmlich der Aufnahme neu erzeugter Clipping-Bereiche.
- **SuperSaveClipRects 0x003C (60)** Ein Adreßzeiger auf die erste *ClipRect*-Datenstruktur einer Liste von Clipping-Bereichen, die im voraus für später als Vorrat allokiert wurden.
- **_cliprects 0x0040 (64)** Ein Adreßzeiger auf eine *ClipRect*-Datenstruktur, der intern von den Systemroutinen während des Refreshs benötigt wird.
- **LayerInfo 0x0044 (68)** Ein Adreßzeiger auf die *LayerInfo*-Datenstruktur des Layerinfos, das diesen Layer enthält.
- **Lock 0x0048 (72)** Eine *SignalSemaphore*-Datenstruktur (siehe »Exec«), die zum Sperren des Layers benötigt wird und dabei als Synchronisationselement für die betroffenen Tasks dient.
- **reserved3 0x0076 (118)** Ein für den Systemgebrauch reserviertes Feld von acht Bytes.
- **ClipRegion 0x007E (126)** Ein Adreßzeiger auf eine *Region*-Datenstruktur, die die Ausmaße und Lager der einzelnen rechteckigen Clipping-Bereiche beschreibt.
- **saveClipRects 0x0082 (130)** Ein Adreßzeiger auf eine *Region*-Datenstruktur, die im Falle von schwerwiegenden (Speicher-)Problemen beim Ausstieg durch die Hintertür von den Systemroutinen verwendet wird, um zu retten, was zu retten ist.
- **reserved2 0x0086 (134)** Ein nicht mehr benutztes Feld von 22 Byte, das aus systeminternen Gründen nicht wegfallen darf.
- **DamageList 0x009C (156)** Ein Adreßzeiger auf eine *Region*-Datenstruktur, über die die eigentliche Damageliste des Layers verwaltet wird. Diese enthält genaue Auskünfte darüber, welche Teile des Layers beim Refresh überhaupt einer Auffrischung bedürfen. Insbesondere im *SimpleRefresh*-Modus kann man diesen Zeiger zur Erzeugung interessanter grafischer Effekte einsetzen.

LayerInfo

Offset		C (graphics/layers.h)	Modula-2 (Graphics)
Hex	Dez	struct Layer_Info	LayerInfo =
		{	RECORD
0000	0	struct Layer *top_layer;	layer:LayerPtr;
0004	4	struct Layer *check_lp;	lp:LayerPtr;
0008	8	struct Layer *obs;	obs:LayerPtr;
000C	12	struct MinList FreeClipRects;	freeClipRects:MinList;
0018	24	struct SignalSemaphore Lock;	lock:SignalSemaphore;
0046	70	struct List gs_Head;	head:List;
0054	84	LONG longreserved;	longreserved:LONGINT;
0058	88	UWORD Flags;	flags:LayerFlagSet;
005A	90	BYTE fatten_count;	count:UByte;
005B	91	BYTE LockLayersCount;	lockLayersCount:UByte;
005C	92	UWORD LayerInfo_extra_size;	layerInfoExtraSize:CARDINAL;
005E	94	WORD *blitbuff;	blitbuff:ADDRESS;
0062	98	struct LayerInfo_extra *LayerInfo_extra;	layerInfoExtra:ADDRESS;
0066	102	};	END;

Assembler — Präfix: *li_*
Syntax: Strukturname = *Layer_Info*
longreserved —> li_long_reserved

Datenstrukturenreferenz — »Graphics«: *Layer.*
»Intuition«: *Screen.*

Routinenreferenz — »Layers«: *CreateBehindLayer, CreateUpfrontLayer, DisposeLayerInfo, FattenLayerInfo, InitLayers, LockLayerInfo, LockLayers, NewLayerInfo, ThinLayerInfo, UnlockLayerInfo, UnlockLayers, WhichLayer.*

Beschreibung

Die *LayerInfo*-Datenstruktur wird von den Systemroutinen der »layers.library« benötigt, um ein reibungsloses Zusammenspiel der verschiedenen Layer eines Viewports bzw. einer Bitmap garantieren zu können.

- **top_layer 0x0000 (0)** — Ein Adreßzeiger auf die *Layer*-Datenstruktur des obersten Layers der zu diesem Layerinfo gehörigen Layerliste.
- **check_lp 0x0004 (4)** — Ein von den Systemroutinen zu Identifikationzwecken intern verwendeter Adreßzeiger auf eine *Layer*-Datenstruktur.
- **obs 0x0008 (8)** — Ein von den Systemroutinen zu Markierungszwecken intern verwendeter Adreßzeiger auf eine *Layer*-Datenstruktur.
- **FreeClipRects 0x000C (12)** — Eine *MinList*-Datenstruktur (siehe »Exec«), über die Adreßzeiger auf freie Clipping-Rechtecke verwaltet werden.

- **Lock 0x0018 (24)** Eine *SignalSemaphore*-Datenstruktur (siehe »Exec«), die zum Sperren des Layerinfos benötigt wird und dabei als Synchronisationselement für die betroffenen Tasks dient.
- **gs_Head 0x0046 (70)** Eine für den Systemgebrauch reservierte *List*-Datenstruktur (siehe »Exec«).
- **longreserved 0x0054 (84)** Ein für den Systemgebrauch reserviertes Langwort.
- **Flags 0x0058 (88)** Eine 2-Byte-Marke, deren Bit 0 anzeigt, ob diesem Layerinfo bereits ein *LayerInfo_extra*-Datenbereich zugeordnet wurde. In diesem Fall ist der Wert von Flags dann ungerade. Zur Benutzung sind folgende Konstanten vordefiniert (die Modula-2-Definition als *LayerFlagSet* ist falsch!):

C	Modula-2	Hex	Dez
NEWLAYERINFO_CALLED	newLayerInfoCalled	0001	1

5

- **fatten_count 0x005A (90)** Ein 1-Byte-Zähler, dessen Wert die Anzahl der Anforderungen wiedergibt, die bisher für die Zuordnung eines *LayerInfo_extra*-Datenbereichs zu diesem Layerinfo gestellt wurden.
- **LockLayersCount 0x005B (91)** Ein 1-Byte-Zähler, dessen Wert gleich der Anzahl der momentan gesperrten Layer ist, die zur Layerliste dieses Layerinfos gehören.
- **LayerInfo_extra_size 0x005C (92)** Eine positive Ganzzahl, deren Wert die Größe des *LayerInfo_extra*-Datenbereichs dieses Layerinfos in Bytes angibt.
- **blitbuff 0x005E (94)** Ein Adreßzeiger auf einen Pufferspeicherbereich, der bei der Durchführung von Blitteroperationen innerhalb der Layer des Layerinfos benötigt wird.
- **LayerInfo_extra 0x0062 (98)** Ein Adreßzeiger auf den *LayerInfo_extra*-Datenbereich dieses Layerinfos, der aus Geschwindigkeitsgründen vom System für diverse Verwaltungszwecke benutzt wird.

RasInfo

Offset		C (graphics/view.h)	Modula-2 (Graphics)
Hex	Dez	struct RasInfo	RasInfo =
		{	RECORD
0000	0	struct RasInfo *Next;	next:RasInfoPtr;
0004	4	struct BitMap *BitMap;	bitMap:BitMapPtr;

Offset		C (graphics/view.h)	Modula-2 (Graphics)
Hex	Dez		
0008	8	SHORT RxOffset;	rxOffset:INTEGER;
000A	10	SHORT RyOffset;	ryOffset:INTEGER;
000C	12	};	END;

Assembler Präfix: *ri_*

Datenstrukturenreferenz »Graphics«: *RasInfo*, *ViewPort*.

Routinenreferenz Keine.

5

Beschreibung

In der *RasInfo*-Datenstruktur werden die Statusinformationen für den Displayspeicher eines Viewport-Playfields untergebracht.

❑ **Next 0x0000 (0)** Ein Adreßzeiger auf eine weitere *RasInfo*-Datenstruktur, die dem zweiten Playfield eines Viewports im Dual-Playfield-Modus zugeordnet ist. Ansonsten enthält diese Komponente immer den Wert *NULL*.

❑ **BitMap 0x0004 (4)** Ein Adreßzeiger auf die der Bitmap des Playfields zugeordnete *BitMap*-Datenstruktur.

❑ **RxOffset 0x0008 (8)** Eine Ganzzahl, deren Wert den horizontalen Versatz des Playfields relativ zu den Grenzen seiner Bitmap in Pixeln angibt.

❑ **RyOffset 0x000A (10)** Eine Ganzzahl, deren Wert den vertikalen Versatz des Playfields relativ zu den Grenzen seiner Bitmap in Pixeln angibt.

RastPort

Offset		C (graphics/rastport.h)	Modula-2 (Graphics)
Hex	Dez	struct RastPort	RastPort =
		{	RECORD
0000	0	struct Layer *Layer;	layer:LayerPtr;
0004	4	struct BitMap *BitMap;	bitMap:BitMapPtr;
0008	8	USHORT *AreaPtrn;	areaPtrn:ADDRESS;
000C	12	struct TmpRas *TmpRas;	tmpRas:TmpRasPtr;
0010	16	struct AreaInfo *AreaInfo;	areaInfo:AreaInfoPtr;
0014	20	struct GelsInfo *GelsInfo;	gelsInfo:GelsInfoPtr;
0018	24	UBYTE Mask;	mask:UByte;

Offset		C (graphics/rastport.h)	Modula-2 (Graphics)
Hex	Dez		
0019	25	BYTE FgPen;	fgPen:UByte;
001A	26	BYTE BgPen;	bgPen:UByte;
001B	27	BYTE AOlPen;	aOlPen:UByte;
001C	28	BYTE DrawMode;	drawMode:DrawModeSet;
001D	29	BYTE AreaPtSz;	areaPtSz:UByte;
001E	30	BYTE linpatcnt;	linPatCnt:UByte;
001F	31	BYTEdummy;	dummy:BYTE;
0020	32	USHORT Flags;	flags:RastPortFlagSet;
0022	34	USHORT LinePtrn;	linePtrn:CARDINAL;
0024	36	SHORT cp_x;	x:INTEGER;
0026	38	SHORT cp_y;	y:INTEGER;
0028	40	UBYTE minterms[8];	minterms:ARRAY [0..7] OF Byte;
0030	48	SHORT PenWidth;	penWidth:INTEGER;
0032	50	SHORT PenHeight;	penHeight:INTEGER;
0034	52	struct TextFont *Font;	font:TextFontPtr;
0038	56	UBYTE AlgoStyle;	algoStyle:FontStyleSet;
0039	57	UBYTE TxFlags;	txFlags:FontFlagSet;
003A	58	UWORD TxHeight;	txHeight:CARDINAL;
003C	60	UWORD TxWidth;	txWidth:CARDINAL;
003E	62	UWORD TxBaseline;	txBaseline:CARDINAL;
0040	64	WORD TxSpacing;	txSpacing:INTEGER;
0042	66	APTR *RP_User;	user:ADDRESS;
0046	70	ULONG longreserved[2];	longreserved: ARRAY [0..1] OF LONGINT;
004E	78	UWORD wordreserved[7];	wordreserved:ARRAY [0..6] OF WORD;
005C	92	UBYTE reserved[8];	reserved:ARRAY [0..7] OF BYTE;
0064	100	};	END;

Assembler

Präfix: *rp_*
Syntax: *dummy —> rp_Dummy*

Datenstrukturenreferenz

»Graphics«: *IODRPReq*, *Layer*.
»Intuition«: *GListEnv*, *IntuitionBase*, *Screen*, *Window*.

Routinenreferenz

»GfxMacros«: *AreaCircle*, *BNDRYOFF*, *DrawCircle*, *SetAfPen*, *SetDrPt*, *SetOPen*, *SetWrMsk*.
»Graphics«: *AddAnimOb*, *AddBob*, *AddVSprite*, *Animate*, *AreaDraw*, *AreaEllipse*, *AreaEnd*, *AreaMove*, *AskFont*, *AskSoftStyle*, *BltBitMapRastPort*, *BltMask-*

5

BitMapRastPort, BltPattern, BltTemplate, ClearEOL, ClearScreen, ClipBlit, DoCollision, Draw, DrawEllipse, DrawGList, Flood, FreeGBuffers, GetGBuffers, InitRastPort, Move, PolyDraw, ReadPixel, RectFill, RemIBob, ScrollRaster, SetAPen, SetBPen, SetDrMd, SetFont, SetRast, SetSoftStyle, SortGList, SwapBitsRastPortClipRect, Text, TextLength, WritePixel.

»Intuition«: *DrawBorder, DrawImage, PrintIText.*

»Layers«: *SwapBitsRastPortClipRect.*

Beschreibung

Die *RastPort*-Datenstruktur dient der Steuerung der grafischen Zeichen- und Animationsroutinen sowie deren Koordination mit dem eigentlichen Displayspeicher der Bitplanes. Deshalb ist jeder Rastport immer einer ganz bestimmten Bitmap zugeordnet, in deren Bitplanes die rufenden Grafikroutinen zeichnen. Alle höheren Displayelemente besitzen auf dem Amiga ihren eigenen *RastPort*-Datensatz. Die überragende Bedeutung dieser Datenstruktur für den reibungslosen Ablauf des Augabegeschehens erkennen Sie nicht zuletzt daran, wie viele Systemroutinen auf einen Rastport direkt zugreifen.

Der Umfang der *RastPort*-Datenstruktur spiegelt die Tatsache wider, daß hier sämtliche Statuswerte des Ausgabezustands eines Displayelements wie z.B. Zeichenmodus oder Füllmuster gespeichert werden.

- **Layer 0x0000 (0)** — Ein Adreßzeiger auf die diesem Rastport zugeordnete *Layer*-Datenstruktur.
- **BitMap 0x0004 (4)** — Ein Adreßzeiger auf die *BitMap*-Datenstruktur, der dieser Rastport zugeordnet ist.
- **AreaPtrn 0x0008 (8)** — Ein Adreßzeiger auf eine Tabelle von Datenworten, die das aktuelle Füllmuster des Rastports darstellen. Die Länge der Tabelle entspricht dem Wert der *AreaPtSz*-Komponente des Rastports.
- **TmpRas 0x000C (12)** — Ein Adreßzeiger auf die dem Rastport momentan zugeordnete *TmpRas*-Datenstruktur, die von den grafischen Füllroutinen benötigt wird.
- **AreaInfo 0x0010 (16)** — Ein Adreßzeiger auf die aktuelle *AreaInfo*-Datenstruktur des Rastports, die von den *Areaxxx*-Routinen zum Füllen von Polygonen verwendet wird.
- **GelsInfo 0x0014 (20)** — Ein Adreßzeiger auf die *GelsInfo*-Datenstruktur, die dem Rastport gerade zugeordnet ist. Über diese Datenstruktur werden sämtliche Animationsobjekte des Rastports verwaltet.
- **Mask 0x0018 (24)** — Eine 1-Byte-Maske, deren einzelne Bits festlegen, welche Bitplanes des Rastports durch die Grafikroutinen tatsächlich

verändert werden. Ein Wert von %00000101 = $05 = 5 für diese Schreibmaske wählt die Bitplanes 0 und 2 aus, wobei alle übrigen desaktiviert werden.

- **FgPen 0x0019 (25)** Eine positive Ganzzahl, deren Wert die Nummer der beim Zeichnen in diesem Rastport verwendeten Vordergrundfarbe angibt.
- **BgPen 0x001A (26)** Eine positive Ganzzahl, deren Wert die Nummer der beim Zeichnen in diesem Rastport verwendeten Hintergrundfarbe angibt.
- **AOlPen 0x001B (27)** Eine Ganzzahl, deren Wert die Farbnummer des »area outline pen« angibt. Diese Farbe wird zum Zeichnen der Umrandung einer gefüllten Fläche benutzt, falls das *AREA-OUTLINE*-Bit in der *Flags*-Komponente des Rastports gesetzt ist.
- **DrawMode 0x001C (28)** Eine 1-Byte-Marke, deren einzelne Bits den aktuellen Zeichenmodus des Rastports festlegen (siehe *DrawModes*).
- **AreaPtSz 0x001D (29)** Eine Ganzzahl, deren Wert die Höhe des aktuellen Füllmusters des Rastports angibt. Dieser Wert sollte einer echten Zweierpotenz entsprechen.
- **linPatCnt 0x001E (30)** Eine von den Zeichenroutinen verwendete positive Ganzzahl, die den aktuellen »Preshift« für das Linienmuster des Rastports festlegt.
- **dummy 0x001F (31)** Ein bisher unbenutztes Füllbyte.
- **Flags 0x0020 (32)** Eine 2-Byte-Marke, deren einzelne Bits das Verhalten der Zeichenroutinen in diesem Rastport steuern. Zur Benutzung sind dabei die folgenden Werte vordefiniert:

C	Modula-2	Hex	Dez	Bedeutung
FRST_DOT	firstDot	0001	1	Ersten Punkt der Linie zeichnen?
ONE_DOT	oneDot	0002	2	Linien im Ein-Punkt-Modus zeichnen?
DBUFFER	dBuffer	0004	4	Ist der Rastport doppelt gepuffert?
AREAOUTLINE	areaOutline	0008	8	Umrandung beim Füllen hervorheben?
NOCROSSFILL	noCrossFill	0020	32	Keine Überschneidungen beim Füllen?

Wichtig: In Modula ist hier der Mengentyp *RastPortFlagSet* zu benutzen.

5

- **LinePtrn 0x0022 (34)** Ein Datenwort, das von den Zeichenroutinen als aktuelles Linienmuster des Rastports verwendet wird.

- **cp_x 0x0024 (36)** Eine postive Ganzzahl, deren Wert die x-Koordinate der aktuellen Position des Grafikcursors innerhalb des Rastports festlegt.

- **cp_y 0x0026 (38)** Eine positive Ganzzahl, deren Wert die y-Koordinate der aktuellen Position des Grafikcursors innerhalb des Rastports festlegt.

- **minTerms 0x0028 (40)** Ein Feld von acht 1-Byte-Marken, die den mit Rastports arbeitenden Blitterroutinen als Minterme der einzelnen Bitplanes des Rastports dienen.

- **PenWidth 0x0030 (48)** Eine positive Ganzzahl, deren Wert die Breite des von den Grafikroutinen verwendeten Zeichenstifts in Pixeln angibt. (**Wichtig**: Wird bisher nicht verwendet!)

- **PenHeight 0x0032 (50)** Eine positive Ganzzahl, deren Wert die Höhe des von den Grafikroutinen verwendeten Zeichenstifts in Pixeln angibt. (**Wichtig**: Wird bisher nicht verwendet!)

- **Font 0x0034 (52)** Ein Adreßzeiger auf die dem Rastport momentan zugeordnete **TextFont**-Datenstruktur, die den für Textausgaben innerhalb des Rastports verwendeten Textfont auswählt.

- **AlgoStyle 0x0038 (56)** Eine 1-Byte-Marke, deren einzelne Bits den algorithmisch zu generierenden Stiltyp für Textausgaben innerhalb des Rastports angeben (siehe *FontStyles*).

- **TxFlags 0x0039 (57)** Eine 1-Byte-Marke, deren Wert Statusinformationen bezüglich des innerhalb des Rastports momentan verwendeten Textfonts enthält.

- **TxHeight 0x003A (58)** Eine positive Ganzzahl, deren Wert die Höhe der Zeichen des momentan innerhalb des Rastports verwendeten Textfonts in Pixeln angibt.

- **TxWidth 0x003C (60)** Eine positive Ganzzahl, deren Wert die nominelle Durchschnittsbreite der Zeichen des momentan innerhalb des Rastports verwendeten Textfonts in Pixeln angibt.

- **TxBaseline 0x003E (62)** Eine Ganzzahl, deren Wert den Abstand (in Pixeln) der Basislinie der Zeichen des momentan innerhalb des Rastports verwendeten Textfonts festlegt. Dieser Abstand wird relativ zum obersten Pixel der Punktmatrix eines jeden Zeichens gemessen.

- **TxSpacing 0x0040 (64)** Eine positive Ganzzahl, deren Wert die Breite der im Rastport auszugebenden Zeichen in Pixeln festlegt. Dieser Wert muß keineswegs dem Wert der *TxWidth*-Komponente entsprechen!

❑ **RP_User 0x0042 (66)** — Ein Adreßzeiger auf einen beliebigen Datenbereich, der diesem Rastport vom Benutzer zur freien Verfügung zugeordnet wurde.

❑ **longReserved 0x0046 (70)** — Ein für den internen Systemgebrauch reserviertes Feld von zwei Langworten.

❑ **wordReserved 0x004E (78)** — Ein nicht mehr benötigtes Feld von sieben Datenworten, das unter früheren Systemversionen eine *Node*-Datenstruktur von »Exec« beinhaltete.

❑ **reserved 0x005C (92)** — Unter früheren Systemversionen für zukünftige Erweiterungen reserviertes Feld von acht Bytes.

Rectangle

Offset		C (graphics/gfx.h)	Modula-2 (Graphics)
Hex	**Dez**	struct Rectangle	Rectangle =
		{	RECORD
0000	0	SHORT MinX;	minX:INTEGER;
0002	2	SHORT MinY;	minY:INTEGER;
0004	4	SHORT MaxX;	maxX:INTEGER;
0006	6	SHORT MaxY;	maxY:INTEGER;
0008	8	};	END;

Assembler — Präfix: *ra_*

Datenstrukturenreferenz — »Graphics«: *ClipRect, Layer, Region, RegionRectangle.*

Routinenreferenz — »Graphics«: *AndRectRegion, ClearRectRegion, OrRectRegion, XorRectRegion.*

Beschreibung

Die *Rectangle*-Datenstruktur dient der Beschreibung beliebiger rechteckiger Displayausschnitte.

❑ **MinX 0x0000 (0)** — Eine Ganzzahl, deren Wert die x-Koordinate des linken oberen Eckpunkts des Rechtecks angibt.

❑ **MinY 0x0002 (2)** — Eine Ganzzahl, deren Wert die y-Koordinate des linken oberen Eckpunkts des Rechtecks angibt.

❑ **MaxX 0x0004 (4)** — Eine Ganzzahl, deren Wert die x-Koordinate des rechten unteren Eckpunkts des Rechtecks angibt.

❑ **MaxY 0x0006 (6)** — Eine Ganzzahl, deren Wert die y-Koordinate des rechten unteren Eckpunkts des Rechtecks angibt.

Region

Offset		C (graphics/regions.h)	Modula-2 (Graphics)
Hex	Dez	struct Region {	Region = RECORD
0000	0	struct Rectangle bounds;	bounds:Rectangle;
0004	4	struct RegionRectangle *RegionRectangle;	regionRectangle: RegionRectanglePtr;
0008	8	};	END;

Assembler	Präfix:	*rg_*
Datenstrukturenreferenz	»Graphics«:	*Layer.*
	»Intuition«:	*IntuitionBase.*
Routinenreferenz	»Graphics«:	*AndRectRegion, AndRegionRegion, ClearRectRegion, ClearRegion, DisposeRegion, NewRegion, OrRectRegion, OrRegionRegion, XorRectRegion, XorRegionRegion.*
	»Layers«:	*InstallClipRegion.*

Beschreibung

Die *Region*-Datenstruktur dient in erster Linie dem Aufbau der Damageliste eines Layers. Zu diesem Zweck ist jeder *Region*-Datenstruktur als Hauptkomponente eine Liste von *RegionRectangle*-Rechtecken zugeordnet. Die Vereinigungsmenge dieser Rechtecke stellt nun genau den von der Region beschriebenen (heterogenen!) Displaybereich dar. Im Falle der Damageliste eines Layers entspricht dieser Bereich gerade den aufzufrischenden Rechteckteilen des Layers.

❑ **bounds 0x0000 (0)**

Eine *Rectangle*-Datenstruktur, die den kleinsten rechteckigen Displayausschnitt beschreibt, in dem sämtliche Rechtecke der Rechteckliste dieser Region vorhanden sind.

❑ **RegionRectangle 0x0004 (4)**

Ein Adreßzeiger auf eine *RegionRectangle*-Datenstruktur, die das Anfangselement der Rechteckliste dieser Region darstellt.

RegionRectangle

Offset Hex	Dez	C (graphics/regions.h)	Modula-2 (Graphics)
		struct RegionRectangle	RegionRectangle =
		{	RECORD
0000	0	struct RegionRectangle *Next;	next:RegionRectanglePtr;
0004	4	struct RegionRectangle *Prev;	prev:RegionRectanglePtr;
0008	8	struct Rectangle bounds;	bounds:Rectangle;
0010	16	};	END;

Assembler Präfix: *rr_*

Datenstrukturenreferenz »Graphics«: *Region, Rectangle.*

Routinenreferenz Keine.

5

Beschreibung

Die *RegionRectangle*-Datenstruktur dient dem Aufbau der Rechteckliste einer *Region*-Datenstruktur.

- **Next 0x0000 (0)** — Ein Adreßzeiger auf die *RegionRectangle*-Datenstruktur des nachfolgenden Elements in der Rechteckliste. Über die *Next*-Zeiger werden sämtliche Rechtecke einer Region vorwärts verkettet.
- **Prev 0x0004 (4)** — Ein Adreßzeiger auf die *RegionRectangle*-Datenstruktur des vorangehenden Elements in der Rechteckliste. Über die *Prev*-Zeiger werden sämtliche Rechtecke einer Region rückwärts verkettet.
- **bounds 0x0008 (8)** — Eine *Rectangle*-Datenstruktur, deren einzelne Komponenten die Lage und Ausmaße des durch diese *RegionRectangle*-Datenstruktur beschriebenen Rechtecks angeben.

SimpleSprite

Offset Hex	Dez	C (graphics/sprite.h)	Modula-2 (Graphics)
		struct SimpleSprite	SimpleSprite =
		{	RECORD
0000	0	UWORD *posctldata;	posctldata:ADDRESS;
0004	4	UWORD height;	height:CARDINAL;
0006	6	UWORD x;	x:CARDINAL;
0008	8	UWORD y;	y:CARDINAL;
000A	10	UWORD num;	num:INTEGER;
000C	12	};	END;

Assembler	Präfix:	*ss_*
Datenstrukturenreferenz	»Intuition«:	*IntuitionBase.*
Routinenreferenz	»Graphics«:	*ChangeSprite, GetSprite, MoveSprite.*

Beschreibung

Die *SimpleSprite*-Datenstruktur wird von den Animationsroutinen zur Darstellung und Verwaltung eines einfachen Hardwaresprites verwendet. Ein einfaches Hardwaresprite ist genau 16 Pixel breit, wohingegen seine Höhe frei wählbar ist. Sämtliche Sprites werden immer im normalen Low-Resolution-Modus dargestellt. Gegenüber virtuellen Sprites und Bobs sind die Möglichkeiten der Animation mit einfachen Hardwaresprites stark eingeschränkt.

❑ **posctldata 0x0000 (0)**

Ein Adreßzeiger auf die eigentlichen Displaydaten des einfachen Sprites, die man auch *SpriteData* nennt. Insgesamt umfassen die Spritedaten genau *2*height+4* Datenworte, die im Speicher folgendermaßen anzuordnen sind:

```
posctldata  ->
+-------------------+
| Wort 0            |
+-------------------+     Kontrollworte
| Wort 1            |
+-------------------+
| Wort 2            |
+-------------------+
|                   |

.........}
.........}                Displaydaten
.........}

|                   |
+-------------------+
| Wort 2*height+1   |
+-------------------+
| Wort 2*height+2   |
+-------------------+     Endemarkierungen
| Wort 2*height+3   |
+-------------------+
```

Die ersten beiden werden von der Spritehardware für die Positions- und Statuskontrolle des Sprites benötigt. Deshalb sollten Sie diese am Anfang stets mit Nullen initialisieren. Die nachfolgenden *2*height*-Datenworte enthalten dagegen die eigentlichen Displaydaten. Für jede Zeile des Sprites sind hier zwei aufeinanderfolgende Datenworte nötig, deren korrespondierende Bits die Farbe des entsprechenden Pixels festlegen. Neben der Farbe »Transparent« mit Farbcode $0 = %00 = 0 sind drei weitere Farben auswählbar. Die letzten beiden Datenworte müssen mit Nullen initialisiert werden, da sie der Hardware als Endemarkierung der Spritedaten dienen.

❑ **height 0x0004 (4)**

Eine positive Ganzzahl, deren Wert die Höhe des einfachen Sprites, gemessen in normalen Low-Resolution-Pixeln festlegt.

❑ **x 0x0006 (6)**

Eine positive Ganzzahl, deren Wert die x-Koordinate der aktuellen Position des einfachen Sprites, gemessen in normalen Low-Resolution-Pixel angibt. Dieser Wert ist relativ zum Viewport oder View zu interpretieren, dem das einfache Sprite bei seiner Erzeugung zugeordnet wurde.

❑ **y 0x0008 (8)**

Eine positive Ganzzahl, deren Wert die y-Koordinate der aktuellen Position des einfachen Sprites, gemessen in normalen Low-Resolution-Pixeln, angibt. Dieser Wert ist relativ zum Viewport oder View zu interpretieren, dem das einfache Sprite bei seiner Erzeugung zugeordnet wurde.

❑ **num 0x000A (10)**

Eine positive Ganzzahl, deren Wert gleich der Nummer des Hardwaresprites ist, dem das einfache Sprite bei seiner Erzeugung zugeordnet wurde.

TextAttr

Offset		C (graphics/text.h)	Modula-2 (Graphics)
Hex	**Dez**	struct TextAttr	TextAttr =
		{	RECORD
0000	0	STRPTR ta_Name;	name:ADDRESS;
0004	4	UWORD ta_YSize;	ySize:CARDINAL;
0006	6	UBYTE ta_Style;	style:FontStyleSet;
0007	7	UBYTE ta_Flags;	flags:FontFlagSet;
0008	8	};	END;

Datenstrukturenreferenz

»DiskFont«: *AvailFont.*
»Intuition«: *IntuiText, IntuitionBase, NewScreen, Window.*

Routinenreferenz

»DiskFont«: *OpenDiskFont.*
»Graphics«: *AskFont, OpenFont.*

Beschreibung

Die *TextAttr*-Datenstruktur dient zur Aufnahme der Attribute Name, Höhe und Schrifttyp, die einen Textfont eindeutig charakterisieren.

❑ **ta_Name 0x0000 (0)**

Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, die den Namen des Textfonts (ohne Höhenangabe!) angibt.

❑ **ta_YSize 0x0004 (4)**

Eine positive Ganzzahl, deren Wert die Höhe des Textfonts in Pixeln festlegt.

❑ **ta_Style 0x0006 (6)**

Eine 2-Byte-Marke, deren einzelne FontStyles-Bits dem voreingestellten Darstellungsmodus der Zeichen des Textfonts entsprechen (siehe *FontStyles*).

❑ **ta_Flags 0x0007 (7)**

Eine 2-Byte-Marke, deren einzelne *FontFlags*-Bits Auskunft über voreingestellte Charakteristika und Statuswerte des Textfonts geben (siehe *FontFlags*).

TextFont

Offset Hex	Dez	C (graphics/text.h)	Modula-2 (Graphics)
		struct TextFont	TextFont =
		{	RECORD
0000	0	struct Message tf_Message;	message:Message;
0014	20	UWORD tf_YSize;	ySize:CARDINAL;
0016	22	UBYTE tf_Style;	style:FontStyleSet;
0017	23	UBYTE tf_Flags;	flags:FontFlagSet;
0018	24	UWORD tf_XSize;	xSize:CARDINAL;
001A	26	UWORD tf_Baseline;	baseline:CARDINAL;
001C	28	UWORD tf_BoldSmear;	boldSmear:CARDINAL;
001E	30	UWORD tf_Accessors;	accessors:CARDINAL;
0020	32	UBYTE tf_LoChar;	loChar:CHAR;
0021	33	UBYTE tf_HiChar;	hiChar:CHAR;
0022	34	APTR tf_CharData;	charData:ADDRESS;
0026	38	UWORD tf_Modulo;	modulo:CARDINAL;
0028	40	APTR tf_CharLoc;	charLoc:ADDRESS;
002C	44	APTR tf_CharSpace;	charSpace:ADDRESS;
0030	48	APTR tf_CharKern;	charKern:ADDRESS;
0034	52	};	END;

Datenstrukturenreferenz

»Console«: *ConUnit.*
»DiskFont«: *DiskFontHeader.*
»Graphics«: *GfxBase, RastPort.*
»Intuition«: *Window.*

Routinenreferenz

»DiskFont«: *OpenDiskFont.*
»Graphics«: *AddFont, CloseFont, OpenFont, RemFont, SetFont.*

Beschreibung

In der *TextFont*-Datenstruktur werden alle Informationen und Statusdaten eines Textfonts (Zeichensatzes) gespeichert und verwaltet. Neben den eigentlichen Pixelmatrizen, die in einer speziell gepackten Form vorliegen, finden sich hier diverse Größen-, Positionier- und Offsetangaben für die Textfonts. Die *TextFont*-Datenstruktur ist dabei so konzipiert, daß sämtliche verfügbaren Textfonts bequem in die Systemfontliste integriert und in dieser schnell selektiert werden können.

❑ **tf_Message 0x0000 (0)**

Eine *Message*-Datenstruktur (siehe »Exec«), die beim Entfernen des Zeichensatzes aus der Systemfontliste als »reply

message« dient. Über die *mn_node*-Komponente dieser *Message*-Datenstruktur ist der Textfont in die Systemfontliste eingereiht (siehe *GfxBase*). Der *ln_name*-Zeiger der *mn_node*-Komponente verweist auf den Namen des Textfonts in Form einer normalen C-Zeichenkette.

- **tf_YSize 0x0014 (20)**
  Eine positive Ganzzahl, deren Wert die Höhe der Zeichen des Textfonts in Pixeln angibt.

- **tf_Style 0x0016 (22)**
  Eine 1-Byte-Marke, deren einzelne *FontStyles*-Bits den Darstellungstyp des Textfonts beschreiben.

- **tf_Flags 0x0017 (23)**
  Eine 1-Byte Marke, deren einzelne *FontFlags*-Bits Statusinformationen bezüglich des Design-Typs des Textfonts beinhalten.

- **tf_XSize 0x0018 (24)**
  Eine positive Ganzzahl, deren Wert die nominale (Durchschnitts-)Breite der Zeichen des Textfonts in Pixeln angibt.

- **tf_Baseline 0x001A (26)**
  Eine positive Ganzzahl, deren Wert die Entfernung (in Pixeln) der Basiszeile vom oberen Zeichenrand für jedes Zeichen des Textfonts festlegt.

- **tf_BoldSmear 0x001C (28)**
  Eine positive Ganzzahl, deren Wert die für die (algorithmische) Erzeugung von fetten Zeichen zu verwendende »Verschmierbreite« in Pixeln angibt.

- **tf_Accessors 0x001E (30)**
  Eine positive Ganzzahl, die als Zähler für die diesen Textfont momentan benutzenden Tasks dient. Erst wenn dieser Zähler gleich Null ist, kann der Textfont nach einem entsprechenden Aufruf der »Graphics«-Routine *RemFont* endgültig aus der Systemfontliste entfernt werden.

- **tf_LoChar 0x0020 (32)**
  Eine positive Ganzzahl, deren Wert den ASCII-Code des kleinsten Zeichens angibt, das in diesem Textfont verfügbar ist (für das der Textfont also eine Punktmatrix enthält).

- **tf_HiChar 0x0021 (33)**
  Eine positive Ganzzahl, deren Wert den ASCII-Code des größten Zeichens angibt, das in diesem Textfont verfügbar ist (für das der Textfont also eine Punktmatrix enthält).

- **tf_CharData 0x0022 (34)**
  Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der die Daten der Pixelmatrizen aller Zeichen des Textfonts enthält. Die Pixelmatrizen liegen dabei zeilenweise in Bitform gepackt vor. Allerdings liegen hier nicht etwa die vollständigen Pixelmatrizen nacheinander in Form einer Tabelle vor. Vielmehr werden die korrespondierenden Zeilen sämtlicher Zeichen hintereinander gepackt, angefangen mit der obersten bis hin zu untersten!

- **tf_Modulo 0x0026 (38)**
  Eine positive Ganzzahl, deren Wert den Abstand (in Bytes) der Pixeldaten zweier aufeinanderfolgender Zeilen eines Zeichens im *tf_CharData*-Bereich angibt.

❑ **tf_CharLoc 0x0028 (40)**

Ein Adreßzeiger auf ein Feld von Ganzzahlpaaren, die den genauen Bitoffset und die Bitbreite sämtlicher Zeichen des Textfonts im *tf_CharData*-Bereich beschreiben.

❑ **tf_CharSpace 0x002C (44)**

Ein Adreßzeiger auf ein Feld von Ganzzahlen, deren Werte die tatsächliche Breite der korrespondierenden Zeichen des Textfonts in Pixeln festlegen. Diese Informationen werden nur für einen proportionalen Zeichensatz benötigt. Bei nichtproportionalen Zeichensätzen ist der Wert dieses Adreßzeigers gleich *NULL,* und für alle Zeichen gilt die nominale Breite aus *tf_XSize.*

❑ **tf_CharKern 0x0030 (48)**

Ein Adreßzeiger auf ein Feld von Ganzzahlen, deren Werte für jedes Zeichen des Textfonts den Abstand zwischen erster Spalte des ausgegebenen Zeichenrasters und dem Beginn der eigentlichen Pixelinformationen festlegen. Ist der Wert dieses Adreßzeigers gleich *NULL,* so gilt für sämtliche Zeichen des Textfonts ein einheitlicher Abstand von null.

TmpRas

Offset		C (graphics/rastport.h)	Modula-2 (Graphics)
Hex	**Dez**	struct TmpRas	TmpRas =
		{	RECORD
0000	0	BYTE *RasPtr;	rasPtr:ADDRESS;
0004	4	LONG Size;	size:LONGINT;
0008	8	};	END;

Assembler Präfix: *tr_*

Datenstrukturenreferenz »Graphics«: *RastPort.* »Intuition«: *IntuitionBase.*

Routinenreferenz »Graphics«: *InitTmpRas.*

Beschreibung

Die *TmpRas*-Datenstruktur dient zur Aufnahme einer einzigen, von den »Graphics«-Füllroutinen (wie z.B. *AreaCircle*) temporär benötigten Bitplane.

❑ **RasPtr 0x0000 (0)**

Ein Adreßzeiger auf das erste Datenbyte der temporären Bitplane, die aus einem zusammenhängenden Speicherbereich im Chip-RAM bestehen muß.

❑ **Size 0x0004 (4)**

Eine positive lange Ganzzahl, deren Wert die Größe der temporären Bitplane in Bytes angibt.

UCopList

Offset		C (graphics/copper.h)	Modula-2 (Graphics)
Hex	Dez	struct UCopList {	UCopList = RECORD
0000	0	struct UCopList *Next;	next:UCopListPtr;
0004	4	struct CopList *FirstCopList;	firstCopList:CopListPtr;
0008	8	struct CopList *CopList;	copList:CopListPtr;
000C	12	};	END;

Assembler Präfix: *ucl_*

Datenstrukturenreferenz »Graphics«: *UCopList, ViewPort.*

Routinenreferenz »GfxMacros«: *CEND, CINIT, CMOVE, CWAIT.*
»Graphics«: *CBump, CMove, CWait, UCopperListInit.*

Beschreibung

Die *UCopList*-Datenstruktur wird zur Verwaltung von Benutzer-Copperlisten verwendet, die intern aus mehreren Zwischen-Copperlisten bestehen.

- **Next 0x0000 (0)** Ein Adreßzeiger auf die nächste *UCopList*-Datenstruktur der Benutzer-Copperliste. Über die *Next*-Zeiger werden sämtliche (Unter-)Abschnitte einer Benutzer-Copperliste vorwärts verkettet.

- **FirstCopList 0x0004 (4)** Ein Adreßzeiger auf den ersten Eintrag in der zu dieser Benutzer-Copperliste gehörigen Zwischen-Copperliste.

- **CopList 0x0008 (8)** Ein Adreßzeiger auf den gerade bearbeiteten Eintrag in der zu dieser Benutzer-Copperliste gehörigen Zwischen-Copperliste.

View

Offset		C (graphics/view.h)	Modula-2 (Graphics)
Hex	Dez	struct View {	View = RECORD
0000	0	struct ViewPort *ViewPort;	viewPort:ViewPortPtr;
0004	4	struct cprlist *LOFCprList;	lofCprList:CprlistPtr;
0008	8	struct cprlist *SHFCprList;	shfCprList:CprlistPtr;
000C	12	short DyOffset;	dyOffset:INTEGER;

Offset		C (graphics/view.h)	Modula-2 (Graphics)
Hex	Dez		
000E	14	short DxOffset;	dxOffset:INTEGER;
0010	16	UWORD Modes;	modes:ViewModeSet;
0012	18	};	END;

Assembler Präfix: *v_*

Datenstrukturenreferenz »Graphics«: *RastPort.*
»Intuition«: *IntuitionBase.*

Routinenreferenz »Graphics«: *InitView, LoadView, MakeVPort, MrgCop.*
»Intuition«: *ViewAddress.*

5

Beschreibung

In der *View*-Datenstruktur sind sämtliche Informationen gespeichert, die zur Erzeugung des Monitorausgabebilds eines Gesamtdisplays (View) und dessen sonstiger Verwaltung notwendig sind. Ein Umschalten zwischen zwei völlig unterschiedlichen Gesamtdisplays ist auf dem Amiga deshalb so leicht realisierbar, indem man für jedes Display eine eigene *View*-Datenstruktur mitsamt den zugehörigen Viewports und Copperlisten entsprechend initialisieren kann.

❑ **ViewPort 0x0000 (0)** Ein Adreßzeiger auf die zum ersten Viewport des Gesamtdisplays gehörige *ViewPort*-Datenstruktur. Dieser Zeiger ist der Startzeiger einer einfach verketteten Liste von *ViewPort*-Datensätzen aller zum Gesamtdisplay gehörigen Viewports.

❑ **LOFCprList 0x0004 (4)** Ein Adreßzeiger auf eine Copperliste für die Erzeugung des Monitorausgabebildes des Gesamtdisplays. Diese Copperliste wird sowohl im Non-Interlace als auch im Interlace-Modus benutzt. Im Interlace-Modus erzeugt sie das Ausgabebild der ungeraden Zeilen. die Abkürzung »LOF« am Anfang des Bezeichners steht für »long frame«.

❑ **SHFCprList 0x0008 (8)** Ein Adreßzeiger auf eine Copperliste, die der Erzeugung des Monitorausgabebildes der geraden Zeilen des Gesamtdisplays im Interlace-Modus dient. Im Non-Interlace-Modus wird diese Copperliste nicht benutzt! Die Abkürzung »SHF« am Anfang des Bezeichners steht für »short frame«.

❑ **DyOffset 0x000C (12)** Eine Ganzzahl, deren Wert den vertikalen Versatz des Gesamtdisplays relativ zum Standardnullpunkt angibt.

❑ **DxOffset 0x000E (14)** Eine Ganzzahl, deren Wert den horizontalen Versatz des Gesamtdisplays relativ zum Standardnullpunkt angibt.

❑ **Modes 0x0010 (16)** Eine 2-Byte-Marke, deren einzelne *ViewModes*-Bits den aktuellen Darstellungsmodus des Views festlegen.

ViewPort

Offset		C (graphics/view.h)	Modula-2 (Graphics)
Hex	Dez	struct ViewPort	ViewPort =
		{	RECORD
0000	0	struct ViewPort *Next;	next:ViewPortPtr;
0004	4	struct ColorMap *ColorMap;	colorMap:ColorMapPtr;
0008	8	struct CopList *DspIns;	dspIns:CopListPtr;
000C	12	struct CopList *SprIns;	sprIns:CopListPtr;
0010	16	struct CopList *ClrIns;	clrIns:CopListPtr;
0014	20	struct UCopList *UCopIns;	uCopIns:UCopListPtr;
0018	24	SHORT DWidth;	dWidth:INTEGER;
001A	26	SHORT DHeight;	dHeight:INTEGER;
001C	28	SHORT DxOffset;	dxOffset:INTEGER;
001E	30	SHORT DyOffset;	dyOffset:INTEGER;
0020	32	UWORD Modes;	modes:ViewModeSet;
0022	34	UBYTE SpritePriorities;	spritePriorities: UByte;
0023	35	UBYTE reserved;	reserved: UByte;
0024	36	struct RasInfo *RasInfo;	rasInfo:RasInfoPtr;
0028	40	};	END;

5

Assembler Präfix: *vp_*

Datenstrukturenreferenz »Graphics«: *CopList, View, ViewPort.*
»Intuition«: *Screen.*

Routinenreferenz »Graphics«: *ChangeSprite, DrawGList, FreeVPortCopLists, InitVPort, LoadRGB4, MakeVPort, MoveSprite, RemIBob, ScrollVPort, SetRGB4, WaitBOVP.*
»Intuition«: *ViewPortAddress.*

Beschreibung

Innerhalb eines Gesamtdisplays, das mittels einer *View*-Datenstruktur verwaltet wird, können beliebige rechteckige Unterdisplayeinheiten eingerichtet werden. Diese ViewPorts werden von der Systemsoftware wie eigenständige Minidisplays verwaltet. Die *ViewPort*-Datenstruktur dient dabei zur Aufnahme der nötigen Statusinformationen wie z.B. Abmessung, Offsets, Farbauswahl, Displaymodus usw.!

Wichtig ist in diesem Zusammenhang noch, daß sich zwei Viewports niemals überlappen dürfen. Weiterhin können Viewports nicht horizontal, sondern immer nur vertikal nebeneinander angeordnet werden, wobei mindestens eine Zeile zwischen ihnen liegen muß.

❑ **Next 0x0000 (0)**

Ein Adreßzeiger auf die *ViewPort*-Datenstruktur, die dem nächsten Viewport in der Viewport-Liste des Gesamtdisplays zugeordnet ist. Der Startzeiger dieser Viewport-Liste steht in der *ViewPort*-Komponente der entsprechenden *View*-Datenstruktur. Über die *Next*-Zeiger werden sämtliche Viewports eines Gesamtdisplays vorwärts verkettet.

5

- **ColorMap 0x0004 (4)** Ein Adreßzeiger auf den *ColorMap*-Datensatz der diesem Viewport zugeordneten Farbtabelle.
- **DspIns 0x0008 (8)** Ein Adreßzeiger auf die Copperliste mit den Anweisungen zur Erstellung des Monitorausgabebildes dieses Viewports.
- **SprIns 0x000C (12)** Ein Adreßzeiger auf die Copperliste mit den Anweisungen zur Erstellung des Monitorausgabebildes der diesem Viewport zugeordneten Sprites.
- **ClrIns 0x0010 (16)** Ein Adreßzeiger auf eine Copperliste, die vom System verwaltet und zur Steuerung der Farbregister dieses Viewports benötigt wird.
- **UCopIns 0x0014 (20)** Ein Adreßzeiger auf eine Benutzer-Copperliste, die Sie nach Belieben zur Erledigung Ihrer ganz persönlichen Aufgaben verwenden können.
- **DWidth 0x0018 (24)** Eine positive Ganzzahl, deren Wert die Höhe des Viewports in Pixeln angibt.
- **DHeight 0x001A (26)** Eine positive Ganzzahl, deren Wert die Breite des Viewports in Pixeln angibt.
- **DxOffset 0x001C (28)** Der vertikale Versatz der linken, oberen Ecke des Viewports relativ zur linken, oberen Ecke des übergeordneten Gesamtdisplays, wobei in Pixeln gemessen wird.
- **DyOffset 0x001E (30)** Der horizontale Versatz der linken, oberen Ecke des Viewports relativ zur linken, oberen Ecke des übergeordneten Gesamtdisplays, wobei in Pixeln gemessen wird.
- **Modes 0x0020 (32)** Eine 2-Byte-Marke, deren einzelne *ViewModes*-Bits den aktuellen Darstellungsmodus des Viewports festlegen.
- **SpritePriorities 0x0022 (34)** Eine 1-Byte-Marke, deren einzelne Bits die Prioritätszustände der diesem Viewport zugeordneten Sprites widerspiegeln.
- **reserved 0x0023 (35)** Ein für den Systemgebrauch bzw. spätere Erweiterungen reserviertes Datenbyte.
- **RasInfo 0x0024 (36)** Ein Adreßzeiger auf die zu diesem Viewport gehörige *RasInfo*-Datenstruktur.

VSprite

Offset		C (graphics/gels.h)	Modula-2 (Graphics)
Hex	Dez	struct VSprite	VSprite =
		{	RECORD
0000	0	struct VSprite *NextVSprite;	nextVSprite:VSpritePtr;
0004	4	struct VSprite *PrevVSprite;	prevVSprite:VSpritePtr;
0008	8	struct VSprite *DrawPath;	drawPath:VSpritePtr;
000C	12	struct VSprite *ClearPath;	clearPath:VSpritePtr;
0010	16	WORD OldY;	oldY:INTEGER;
0012	18	WORD OldX;	oldX:INTEGER;
0014	20	WORD Flags;	flags:VSpriteFlagSet;
0016	22	WORD Y;	y:INTEGER;
0018	24	WORD X;	x:INTEGER;
001A	26	WORD Height;	height:INTEGER;
001C	28	WORD Width;	width:INTEGER;
001E	30	WORD Depth;	depth:INTEGER;
0020	32	WORD MeMask;	meMask:BITSET;
0022	34	WORD HitMask;	hitMask:BITSET;
0024	36	WORD *ImageData;	imageData:ADDRESS;
0028	40	WORD *BorderLine;	borderLine:ADDRESS;
002C	44	WORD *CollMask;	collMask:ADDRESS;
0030	48	WORD *SprColors;	sprColors:ADDRESS;
0034	52	struct Bob *VSBob;	vsBob:BobPtr;
0038	56	BYTE PlanePick;	planePick:UByte;
0039	57	BYTE PlaneOnOff;	planeOnOff:UByte;
003A	58	VUserStuff VUserExt;	
003C	60	};	END;

Assembler — Präfix: *vs_*
Syntax: Strukturname = *VS*
VUserExt —> *vs_SUserExt*

Datenstrukturenreferenz — »Graphics«: *Bob*, *DBufPacket*, *GelsInfo*, *VSprite*.

Routinenreferenz — »Graphics«: *AddVSprite*, *InitGels*, *InitMasks*, *RemV Sprite*.

Beschreibung

Die *VSprite*-Datenstruktur stellt die Basis des gesamten Animationssystems der Amiga-Systemsoftware dar. Die *VSprite*-Datenstruktur ist der Grundbaustein aller verfügbaren Datenstrukturen zur Beschreibung von Animationselementen. Dies geht besonders deutlich aus der Beschreibung zur *AnimOb*-Datenstruktur hervor!

Wie das gesamte Animationssystem hat auch die *VSprite*-Datenstruktur ein geteiltes Wesen, das sich sowohl aus speziellen Sprite- und Bob-Anteilen als auch aus diesen beiden Arten von GELs gemein-

samen Anteilen zusammensetzt. Isoliert betrachtet dient die *VSprite*-Datenstruktur zunächst nur der Repräsentation virtueller Sprites. Diese sind eine Erweiterung der Hardwaresprites auf Software-Basis. Die virtuelle Spritemaschine kann beliebig viele virtuelle Sprites gleichzeitig verwalten und auf dem Bildschirm bewegen. Darüber hinaus bestehen die zusätzlichen Eigenschaften virtueller Sprites in eigenen Farbpaletten sowie diversen Möglichkeiten der (Software-)Kollisionserkennung und -behandlung.

Zusammen mit einer *Bob*-Datenstruktur dient die *VSprite*-Datenstruktur der Repräsentation eines Bobs. Ein Bob kann ohne *VSprite*-Datenstruktur nicht existieren!

- **NextVSprite 0x0000 (0)** — Ein Adreßzeiger auf die *VSprite*-Datenstruktur des nächsten GELs in der GEL-Liste des Rastports. Über die *NextVSprite*-Zeiger werden sämtliche GELs der GEL-Liste eines Rastports vorwärts verkettet.

- **PrevVSprite 0x0004 (4)** — Ein Adreßzeiger auf die *VSprite*-Datenstruktur des vorangehenden GELs in der GEL-Liste des Rastports. Über die *PrevVSprite*-Zeiger werden sämtliche GELs der GEL-Liste eines Rastports rückwärts verkettet.

- **DrawPath 0x0008 (8)** — Ein Adreßzeiger auf die *VSprite*-Datenstruktur des Bobs, das vom Animationssystem nach diesem gezeichnet wird. Über die *DrawPath*-Zeiger werden die *VSprite*-Datenstrukturen sämtlicher Bobs einer GEL-Liste in Form einer linearen Liste vorwärts verkettet. Diese Liste dient dem System als Pfad beim tatsächlichen Zeichnen der Bobs. Die Anordnung der Bobs in der *DrawPath*-Liste wird vom Animationssystem beim Aufruf der »Graphics«-Routine *DrawGList* anhand der *After*- und *Before*-Zeichenprioritäten der einzelnen Bobs bestimmt.

- **ClearPath 0x000C (12)** — Ähnlich wie *DrawPath*, allerdings beschreibt diese Liste den Löschpfad zum Löschen von Bob-Überlagerungen.

- **OldY 0x0010 (16)** — Eine Ganzzahl, deren Wert die y-Koordinate der letzten Position des GELs relativ zum umgebenden Rastport angibt.

- **OldX 0x0012 (18)** — Eine Ganzzahl, deren Wert die x-Koordinate der letzten Position des GELs relativ zum umgebenden Rastport angibt.

- **Flags 0x0014 (20)** — Eine 2-Byte-Marke, deren einzelne Bits Statusinformationen bezüglich der aktuellen Darstellungsform des GELs enthalten. Zur Benutzung sind dabei die folgenden Konstanten vordefiniert:

C	Modula-2	Hex	Dez	Bedeutung
VSPRITE	vsprite	0001	1	gesetzt=virt. Sprite, gelöscht=Bob
SAVEBACK	saveBack	0002	2	Soll Hintergrund gespeichert werden?

C	Modula-2	Hex	Dez	Bedeutung
OVERLAY	overlay	0004	4	Soll Bob-Image Hintergrund überdecken?
MUSTDRAW	mustDraw	0008	8	Virtuelles Sprite unbedingt zeichnen?
BACKSAVED	backSaved	0100	256	Ist Hintergrund des Bobs gespeichert?
BOBUPDATE	bobUpdate	0200	512	Ist der Bob-Update gerade in Gang?
GELGONE	gelGone	0400	1024	Ist GEL außerhalb des Clip-Bereichs?
VSOVERFLOW	vsOverflow	0800	2048	Ist ein VSprite-Überlauf aufgetreten?

Wichtig: In Modula-2 ist hier der Mengentyp *VSpriteFlagSet* zu verwenden. Die Maske *SUSERFLAGS* = 0x00FF = 255 aller benutzermanipulierbaren Bits der Flags-Maske ist unter C bereits verfügbar. Für Modula-2 ist sie folgendermaßen zu definieren: »*SUserFlags = VSpriteFlagSet{vsprite, saveBack, overlay, mustDraw, vf4, vf5, vf6, vf7)*«!

❑ **Y 0x0016 (22)**

Eine Ganzzahl, deren Wert die y-Koordinate der augenblicklichen Position des GELs relativ zum umgebenden Rastport angibt.

❑ **X 0x0018 (24)**

Eine Ganzzahl, deren Wert die x-Koordinate der augenblicklichen Position des GELs relativ zum umgebenden Rastport angibt.

❑ **Height 0x001A (26)**

Eine Ganzzahl, deren Wert der Höhe des GEL-Images in Pixeln entspricht.

❑ **Width 0x001C (28)**

Eine Ganzzahl, deren Wert der Breite des GEL-Images entspricht. Für echte virtuelle Sprites sollte hier immer der Wert 16 eingetragen sein. Im Falle der *VSprite*-Datenstruktur eines Bobs, gibt dieser Wert die Breite seiner Image-Bitplanes in Datenworten an (siehe *ImageData*-Komponente).

❑ **Depth 0x001E (30)**

Eine Ganzzahl, deren Wert die Tiefe des GEL-Images in Bitebenen angibt. Diese Komponente ist lediglich für die *VSprite*-Datenstruktur eines Bobs nicht jedoch für echte virtuelle Sprites von Bedeutung.

❑ **MeMask 0x0020 (32)**

Eine 2-Byte-Maske, deren einzelne Bits angeben, mit welchen GELs dieses GEL eine Kollision erzeugt, falls es von diesen gerammt wird. In diesem Fall wird bei einem Aufruf der »Graphics«-Routine *DoCollision* die *HitMask* des rammenden und am weitesten links oben liegenden GELs mit der *MeMask* des gerammten und sich rechts unten befindlichen

GELs durch eine logische UND-Operation verknüpft. Die so erhaltene Bitmaske gibt anhand der gesetzten Bits Auskunft darüber, welche Art von Kollision (wenn überhaupt) stattgefunden hat. Zur Kollisionsbehandlung ruft das Animationssystem danach automatisch die entsprechende Kollisionsroutine aus der *GelsInfo*-Datenstruktur der GEL-Liste auf. Sind gleich mehrere Kollisionen auf einmal aufgetreten, so wird lediglich die niederwertigste Kollisionsroutine aktiviert. Wichtig bei der Benutzung dieser beiden Stoßmasken ist, daß außer dem für Randkollisionen reservierten Bit 0 alle anderen Bits für den Benutzer frei verfügbar sind.

- **HitMask 0x0022 (34)**

Eine 2-Byte-Maske, deren einzelne Bits angeben, mit welchen GELs dieses GEL eine Kollision erzeugt, falls es sie rammt (siehe auch *MeMask*-Komponente).

- **ImageData 0x0024 (36)**

Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der die eigentlichen Displaydaten des GEL-Images enthält.

a) Im Falle eines echten virtuellen Sprites umfaßt dieser Bereich genau *2*Height* Datenworte. Der *Image*-Datenbereich eines virtuellen Sprites entspricht im Aufbau den Displaydaten eines Hardwaresprites (siehe *SimpleSprite*). Allerdings entfallen hier die beiden einleitenden und abschließenden Kontrollworte.

b) Das Image eines Bobs wird im ganz normalen Playfield-Format gespeichert: Der *Image*-Datenbereich umfaßt hier genau *Height*Width*Depth*/16 Datenworte und setzt sich aus *Depth* aufeinanderfolgenden Bitplanes der Größe *Height*Width*/16 (in Datenworten!) zusammen.

- **BorderLine 0x0028 (40)**

Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der für ein echtes virtuelles Sprite nur ein Datenwort, im Fall einer *VSprite*-Datenstruktur eines Bobs dagegen genau *Width* Datenworte enthält. Diese Datenworte benutzt das Animationssystem zur schnellen Erkennung von GEL-Kollisionen mit der rechten oder linken Randbegrenzung des Clipping-Bereichs. Der Wert der Datenworte ergibt sich aus einer kumulierten logischen ODER-Verknüpfung sämtlicher Zeilen der Kollisionsmaske des GELs. Dadurch wird erreicht, daß in der *BorderLine*-Maske das der äußersten rechten und der äußersten linken, kollisionssensitiven Stelle des GELs entsprechende Bit gesetzt ist.

- **CollMask 0x002C (44)**

Ein Adreßzeiger auf einen Speicherbereich im Chip-RAM, der genauso groß wie das ganze Image des virtuellen Sprites bzw. wie eine Bitplane des Bob-Images sein muß (im Fall der *VSprite*-Datenstruktur eines Bobs). Die gesetzten Bits dieser Kollisionsmaske geben die Pixel bzw. Stellen des GELs an,

die sensitiv für Kollisionen mit anderen GELs oder den Randbegrenzungen des Clipping-Bereichs sind.

- **SprColors 0x0030 (48)**

Ein Adreßzeiger auf einen mindestens 6 Byte großen Speicherbereich im Chip-RAM, dessen erste drei Datenworte die Farbtabelle dieses virtuellen Sprites darstellen. Ist der Wert des Zeigers gleich *NULL*, so werden zur Darstellung des Sprites keine neuen Farben verwendet. Für die *VSprite*-Datenstruktur eines Bobs ist dieser Zeiger bedeutungslos.

- **VSBob 0x0034 (52)**

Ein Adreßzeiger auf die *Bob*-Datenstruktur des dieser *VSprite*-Datenstruktur übergeordneten Bobs (falls es sich um ein solches handelt). Handelt es sich dagegen um ein echtes virtuelles Sprite, so ist dieser Zeiger bedeutungslos, weswegen man ihn am besten mit dem Wert *NULL* initialisiert.

5

- **PlanePick 0x0038 (56)**

Eine 1-Byte-Maske, deren einzelne Bits festlegen, in welche Bitplanes des Rastports das Bob-Image des zu dieser *VSprite*-Datenstruktur gehörigen Bobs gezeichnet werden soll. Die Nummern der gesetzten Bits entsprechen dabei den Bitplanes, in die gezeichnet werden soll. Ein *PlanePick*-Wert von %00011001 = 0x0019 = 25 wählt beispielsweise die Bitplanes 0, 3 und 4 zum Zeichnen aus. Für echte virtuelle Sprites ist diese Maske ohne Bedeutung.

- **PlaneOnOff 0x0039 (57)**

Eine 1-Byte-Maske, deren einzelne Bits festlegen, was in die Bitplanes des Rastports gezeichnet werden soll, die nicht durch die *PlanePick*-Maske zum Zeichnen des Bob-Images ausgewählt werden. Die Bit-nach-Bitplane Zuordnung entspricht bei der *PlaneOnOff*-Maske derjenigen der *PlanePick*-Maske. Überall dort, wo im *Imageshadow* (siehe *Bob*-Datenstruktur) des Bobs ein Bit gesetzt ist, wird in die nicht durch *PlanePick* ausgewählten Bitplanes der Wert des entsprechenden *PlaneOnOff*-Bits geschrieben. Wie die *PlanePick*-Maske ist auch diese Maske für echte virtuelle Sprites ohne Bedeutung.

- **VUserExt 0x003A (58)**

Eine vom Benutzer völlig frei definierbare Erweiterung der *VSprite*-Datenstruktur zur Aufnahme weiterer Statusinformationen. Zu diesem Zweck ist der Typ *VUserStuff* im Benutzerprogramm entsprechend zu deklarieren. (Wichtig: Diese Erweiterung ist nur unter C, nicht jedoch unter Modula-2 verfügbar!)

5

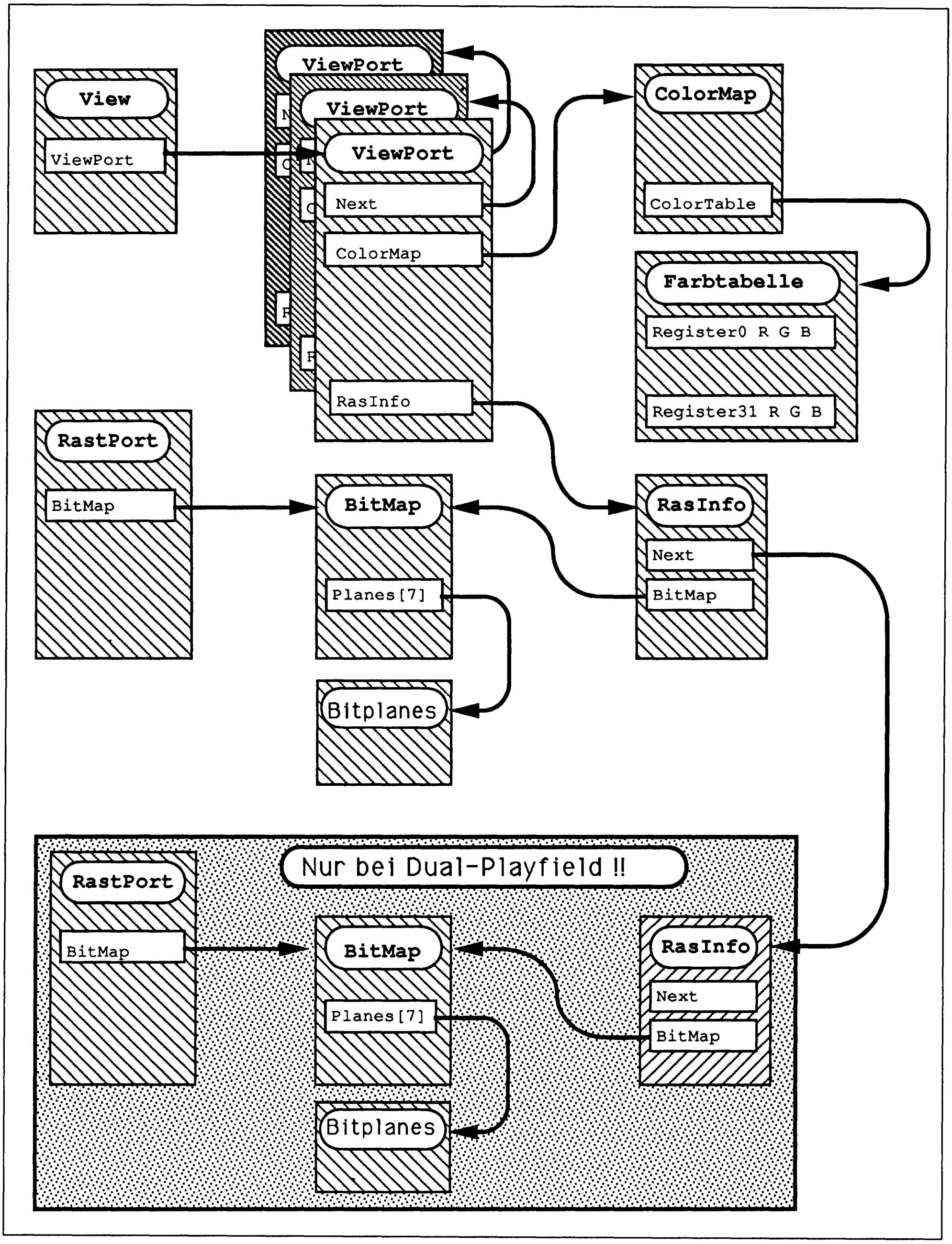


Abb. 5.1: Der Aufbau eines View

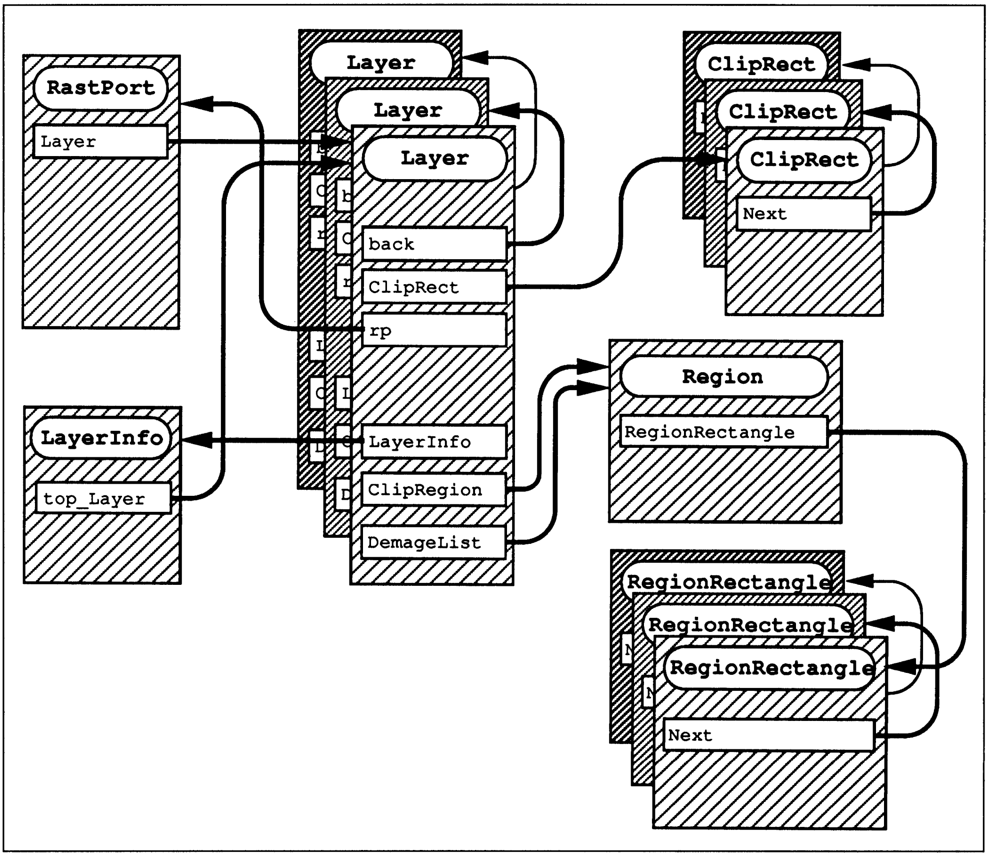


Abb. 5.2: Rastport und Layers

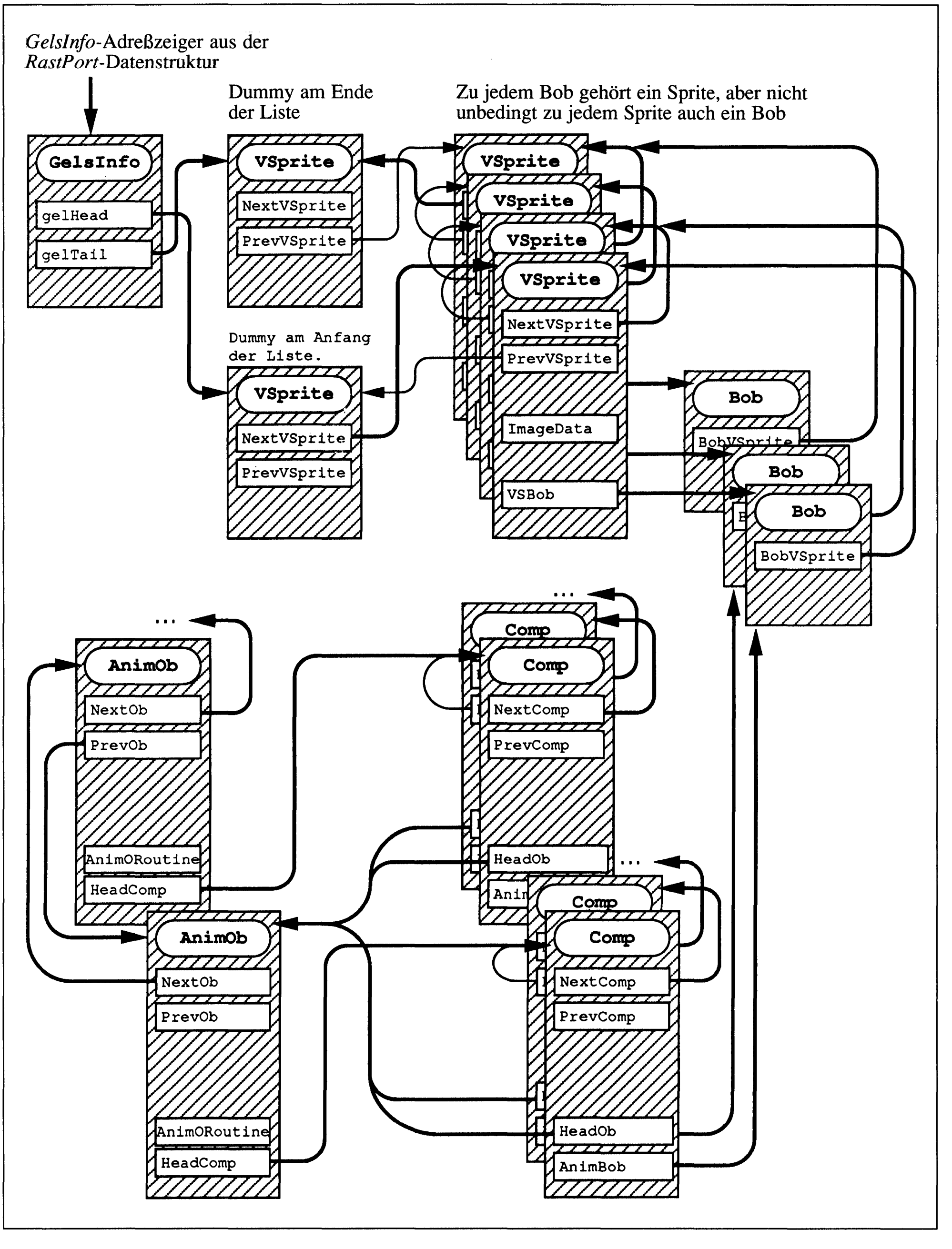


Abb. 5.3: Das Animations-System

6 DIE DOS-LIBRARY

Die DOS-Library ist für die gesamte Ein- und Ausgabe (insbesondere betreffend der Laufwerke) verantwortlich.

6.1 Spezielle Datentypen der DOS-Library

BPTR

Dies ist eine spezielle Art von Adreßzeigern, die nur von DOS verwendet wird. Ein BPTR, der auf ein Objekt zeigt, hat den Wert Adresse durch 4. Falls also das referenzierte Objekt bei 4000 im Speicher liegt, dann hat ein BPTR auf dieses Objekt den Wert 1000. Daraus folgt, daß ein Objekt, auf das ein BPTR zeigt, immer an einer durch 4 teilbaren Adresse liegen muß. Die Adressen fast aller DOS-Datenstrukturen werden als solche BPTR-Adreßzeiger verwaltet.

BSTR

Diese besondere Art der Speicherung von Zeichenketten wird von Amiga-DOS verwendet. Ein BSTR ist ein BPTR auf einen Speicherbereich, in dem im ersten Byte die Länge der Zeichenkette und nachfolgend die Zeichenkette selbst steht. In Modula-2 ist BSTR als *BPOINTER TO ARRAY[0..255] OF CHAR* vereinbart.

Die Segment-Liste

Die im Speicher verstreuten Codestücke eines Programms werden mit Hilfe einer Segment-Liste verknüpft. Die Liste hat die folgende Datenstruktur:

LONG SegLength;
BPTR NextSeg;
CODE

Wie Sie sehen, wird der eigentliche Code von der Längenangabe in *SegLength* und dem BPTR auf das nächste Segment (oder *NULL*) in *NextSeg* eingeleitet.

6.2 Fehlermeldungen der DOS-Library

Fehlermeldungen können nach allen Zugriffen auf Dateien oder auf Teile des »Filesystems« mit Hilfe der *IoErr*-Routine geholt werden. Dabei ist zu beachten, daß das Ergebnis eines *IoErr*-Aufrufes normalerweise nur dann eine gültige Fehlermeldung ist, wenn tatsächlich ein Fehler aufgetreten ist. Folgende Fehlermeldungen können auftreten (zu den Definitionen in C und Assembler gehört jeweils noch das Präfix *ERROR_*):

C	Modula-2	Dez	Beschreibung
NO_FREE_STORE	noFreeStore	103	Nicht genug Speicherplatz frei.
TASK_TABLE_FULL	taskTableFull	105	Kein Platz mehr in der Task-Tasktabelle von RootNode.
LINE_TOO_LONG	lineTooLong	120	Die Eingabezeile ist zu lang.
FILE_NOT_OBJECT	fileNotObject	121	Das File ist kein Objektmodul.
INVALID_RESIDENT _LIBRARY	invalidResident Library	122	Ungültige Resident-Bibliothek beim Laden.
NO_DEFAULT_DIR	noDefaultDir	201	Es ist kein Verzeichnis voreingestellt.
OBJECT_IN_USE	objectInUse	202	Ein anderer Task greift bereits auf den Eintrag zu.
OBJECT_EXISTS	objectExists	203	Der Eintrag existiert bereits.
DIR_NOT_FOUND	dirNotFound	204	Verzeichnis nicht gefunden.
OBJECT_NOT_FOUND	objectNotFound	205	Eintrag nicht gefunden.
BAD_STREAM_NAME	badStreamName	206	Ein/Ausgabestromname ungültig.
OBJECT_TOO_LARGE	objectTooLarge	207	Der Eintrag ist zu groß.
ACTION_NOT_KNOWN	actionNotKnown	209	Der Befehl ist ungültig.
INVALID_COMPONENT NAME	invalidComponent Name	210	Im Namen befindet sich ein ungültiges Zeichen.
INVALID_LOCK	invalidLock	211	Die Lock-Angabe ist ungültig.
OBJECT_WRONG_TYPE	objectWrongType	212	Falscher Objekttyp (z.B. Directory statt File).
DISK_NOT_VALIDATED	diskNotValidated	213	Diskette ist ungültig.
DISK_WRITE_PROTECTED	diskWriteProtected	214	Diskette ist schreibgeschützt.
RENAME_ACROSS_ DEVICES	renameAcross Devices	215	Bei RENAME wurde ein falscher Gerätename angegeben.
DIRECTORY_NOT_EMPTY	directoryNotEmpty	216	Verzeichnis ist nicht leer.
TOO_MANY_LEVELS	tooManyLevels	217	???
DEVICE_NOT_MOUNTED	deviceNotMounted	218	Gerät ist nicht angemeldet.
SEEK_ERROR	seekError	219	Fehler beim Suchzugriff auf eine Datei.
COMMENT_TOO_BIG	commentToBig	220	Der Kommentar ist zu lang.

6

C	Modula-2	Dez	Beschreibung
DISK_FULL	diskFull	221	Die Diskette ist voll.
DELETE_PROTECTED	deleteProtected	222	Löschen der Datei verboten.
WRITE_PROTECTED	writeProtected	223	Datei ist schreibgechützt.
READ_PROTECTED	readProtected	224	Datei ist lesegeschützt.
NOT_A_DOS_DISK	notADosDisk	225	Dies ist keine DOS-Diskette.
NO_DISK	noDisk	226	Es ist keine Diskette eingelegt.
NO_MORE_ENTRIES	noMoreEntries	232	Es sind keine weiteren Einträge vorhanden.

6.3 Die DOS-Packets

6

Der nachfolgenden Auflistung können Sie die Bedeutung einiger *dp_Type*-Werte der *DosPacket*-Datenstruktur und der dazugehörigen Argumente für die DOS-Handler entnehmen. Die DOS-Packets werden zusammen mit einer Nachricht (*Message*-Datenstruktur, siehe »Exec«) an den Port des Handlerprozesses geschickt. Da einige Packets nirgends richtig dokumentiert sind, war bei manchen Parametern auch mit gezielten »Herumhacken« nicht viel herauszufinden...

C	Modula-2	Dez	Beschreibung
ACTION_NIL	nil	0	Tue gar nichts.
ACTION_GET_BLOCK	getBlock	2	Einen Block einlesen.
ACTION_SET_MAP	setMap	4	Setzt eine neue Keymap.
ACTION_DIE	die	5	Für verzweifelte Benutzer oder um Prozeß zu killen ???
ACTION_EVENT	event	6	Irgendwas wird wohl passieren ???
ACTION_CURRENT_ VOLUME	currentVolume	7	???
ACTION_LOCATE_OBJEKT	locateObject	8	Implementiert die *Lock*-Funktion

dp_Arg1: Ein BPTR auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg2: Ein BSTR auf den Eintragsnamen.
dp_Arg3: Der Zugriffmodus (siehe *FileLock*-Datenstruktur).

C	Modula-2	Dez	Beschreibung
ACTION_RENAME_DISK	renameDisk	9	Das aktuelle Volume umbenennen.

dp_Arg1: Ein BSTR-Adreßzeiger auf den neuen Namen.

ACTION_FREE_LOCK	freeLock	15	Implementiert die *Unlock*-Funktion.

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur.

ACTION_DELETE_OBJEKT	deleteObject	16	Implementiert die *Delete*-Funktion.

6

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg2: Der Eintragname als BSTR.

ACTION_RENAME_OBJEKT	renameObject	17	Implementiert die *Rename*-Funktion.

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg2: Der Eintragname als BSTR-Adreßzeiger.
dp_Arg3: Ein BPTR auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der umbenannte Eintrag gehören soll.
dp_Arg4: Der neue Eintragname als BSTR.

ACTION_COPY_DIR	copyDir	19	Implementiert die *DupLock*-Funktion.

dp_Arg1: Ein BPTR-Adreßzeiger auf die zu duplizierende *FileLock*-Datenstruktur.
dp_Res1: Hier steht hinterher ein BPTR-Adreßzeiger auf die neuen *FileLock*-Datenstruktur.

ACTION_WAIT_CHAR	waitChar	20	Implementiert die *WaitForChar*-Funktion

dp_Arg1: Die Zeitspanne, die abgewartet werden soll.

C	Modula-2	Dez	Beschreibung
ACTION_SET_PROTECT	setProtect	21	Implementiert die *SetProtection*-Funktion.

dp_Arg1: Nicht benutzt.
dp_Arg2: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg3: Name des Eintrags als BSTR.
dp_Arg3: Die »Protectionbits«.

ACTION_CREATE	createDir	22	Implementiert die *CreateDir*-Funktion.

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg2: Der Eintragname als BSTR.
dp_Res1: Ein BPTR auf die *FileLock*-Datenstruktur des neuen Verzeichnisses.

6

ACTION_EXAMINE_OBJEKT	examineObject	23	Implementiert die *Examine*-Funktion.

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Eintrags.
dp_Arg2: Ein BPTR-Adreßzeiger auf eine allokierte *FileInfoBlock*-Datenstruktur.

ACTION_EXAMINE_NEXT	examineNext	24	Implementiert die *ExamineNext*-Funktion

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Eintrags.
dp_Arg2: Ein BPTR-Adreßzeiger auf eine *FileInfoBlock*-Datenstruktur.

ACTION_DISK_INFO	diskInfo	25	Implementiert die *DOS-Info*-Funktion.

dp_Arg1: Ein BPTR auf eine allokierte *InfoData*-Datenstruktur.

C	Modula-2	Dez	Beschreibung
ACTION_INFO	info	26	Implementiert den CLI-Befehl *INFO*.
ACTION_FLUSH	flush	27	Flush-Befehl der Device ausführen.
ACTION_SET_COMMENT	setComment	28	Implementiert die SetComment-Funktion.

dp_Arg1: Nicht benutzt.
dp_Arg2: Ein BPTR auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg3: Der Name des Eintrags als BSTR.
dp_Arg3: Zu setzender Kommentar als BSTR.

ACTION_PARENT	parent	29	Implementiert die *ParentDir*-Funktion.

6

dp_Arg1: Ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Eintrages.
dp_Res1: Hinterher ein BPTR auf die *FileLock*-Datenstruktur des »Parent-Directory«.

ACTION_TIMER	timer	30	???
ACTION_INHIBIT	inhibit	31	Die in das Laufwerk des Handlers eingelegte Diskette wird (nicht) geprüft.

dp_Arg1: 1 = Diskette wieder prüfen; 0 = Diskette nicht prüfen.

ACTION_DISK_TYPE	diskType	32	Trackdisk-Device-Befehl *DISKTYPE* ???
ACTION_DISK_CHANGE	diskChange	33	CLI-Befehl *DISKCHANGE* ???
ACTION_SET_DATE	setDate	34	Das Datum eines Eintrags setzen.

dp_Arg1: Nicht benutzt.
dp_Arg2: Ein BPTR auf die *FileLock*-Datenstruktur des Verzeichnisses, zu dem der Eintrag gehört.
dp_Arg3: Der Name des Eintrags als BSTR-Adreßzeiger.
dp_Arg3: Ein BPTR auf eine *DateStamp*-Datenstruktur, die das neue Datum enthält.

C	Modula-2	Dez	Beschreibung
ACTION_WRITE	write	52	Implementiert die *Write*-Funktion.

dp_Arg1: *Arg1*-Feld der zugehörigen *FileHandle*-Datenstruktur.
dp_Arg2: Ein Adreßzeiger auf den Schreibpuffer.
dp_Arg3: Die Anzahl der zu schreibenden Zeichen.
dp_Res1: Enthält hinterher die Anzahl der geschriebenen Zeichen oder bei einem Fehler –1.

C	Modula-2	Dez	Beschreibung
ACTION_READ	read	57	Implementiert die *Read*-Funktion.

dp_Arg1: *Arg1*-Feld der zugehörigen *FileHandle*-Datenstruktur.
dp_Arg2: Ein Adreßzeiger auf den Lesepuffer.
dp_Arg3: Die Länge des Lesepuffers.
dp_Res1: Enthält hinterher die Anzahl der gelesenen Zeichen oder –1 für einen aufgetretenen Lesefehler.

6

6.4 Die Datenstrukturen der DOS-Library

CommandLineInterface

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez	struct CommandLineInterface	CommandLineInterface
		{	RECORD
0000	0	LONG cli_Result2;	result2:LONGINT;
0004	4	BSTR cli_SetName;	setName:BSTR;
0008	8	BPTR cli_CommandDir;	commandDir:PathInfoPtr;
000C	12	LONG cli_ReturnCode;	returnCode:LONGINT;
0010	16	BSTR cli_CommandName;	commandName:BSTR;
0014	20	LONG cli_FailLevel;	failLevel:LONGINT;
0018	24	BSTR cli_Prompt;	prompt:BSTR;
001B	28	BPTR cli_StandardInput;	standardInput:FileHandlePtr;
0020	32	BPTR cli_CurrentInput;	currentInput:FileHandlePtr;
0024	36	BSTR cli_CommandFile;	commandFile:BSTR;
0028	40	LONG cli_Interactive;	interactive:LONGINT;
002B	44	LONG cli_Background;	background:LONGINT;
0030	48	BPTR cli_CurrentOutput;	currentOutput:FileHandlePtr;
0034	52	LONG cli_DefaultStack;	defaultStack:LONGINT;
0038	56	BPTR cli_StandardOutput;	standardOutput:FileHandlePtr;
002B	60	BPTR cli_Module;	module:BPTR;
0030	64	};	END;

Datenstrukturenreferenz »Dos«: *Process.*

Routinenreferenz Keine.

Beschreibung

Wenn ein Prozeß ein CLI-Prozeß ist, dann steht die Adresse dieser Datenstruktur in dem *CLIStruct*-Feld der *Process*-Datenstruktur. Sie enhält einige für das CLI wichtige Informationen wie beispielsweise das aktuelle Verzeichnis oder den letzte Fehlercode.

- **cli_Result2 0x0000 (0)** Hier steht die DOS-Fehlernummer des letzten Befehls. Sie wird von der *IoErr*-Routine geliefert.
- **cli_SetName 0x0004 (4)** Ein BSTR auf den Namen der aktuellen Directories.
- **cli_CommandDir 0x0008 (8)** Ein BPTR auf eine Liste der *FileLock*-Datenstrukturen,aus denen die CLI-Befehle gelesen werden. Jedes Element der Liste besteht aus einem BPTR auf das nächste Element und einem BPTR auf die *FileLock*-Datenstruktur eines Directorys (die mit dem CLI-Befehl *PATH* angegeben wurde).

6

- **cli_ReturnCode 0x000C (12)** Das vom letzten Befehl zurückgegebene Ergebnis. Je nach Schwere des Fehlers kann hier einer der folgenden Werte stehen.

C	Modula-2	Hex	Beschreibung
RETURN_OK	ok	00	Es ist kein Fehler aufgetreten.
RETURN_WARN	warn	05	Der augenblickliche Status ist unsicher.
RETURN_ERROR	error	0A	Ein »normaler« Fehler ist aufgetreten.
RETURN_FAIL	fail	14	Ein schwerwiegender Fehler ist aufgetreten.

- **cli_CommandName 0x0010 (16)** Ein BSTR auf den Namen des aktuellen Befehls.
- **cli_FailLevel 0x0014 (20)** Hier steht der mit dem CLI-Befehl *FAILAT* gesetzte Fehlerlevel. Er bestimmt, ab welchen *cli_ReturnCode* die Kommandofolge unterbrochen wird.
- **cli_Prompt 0x0018 (24)** Ein BSTR auf die Zeichenkette, mit der sich das CLI zur Befehlseingabe meldet (der »Prompt«). Er wird mit dem CLI-Befehl *PROMPT* gesetzt.
- **cli_StandardInput 0x001B (28)** Ein BPTR-Adreßzeiger auf die *FileHandle*-Datenstruktur des voreingestellten Eingabegeräts (In Modula-2 ein *FileHandlePtr*).
- **cli_CurrentInput 0x0020 (32)** Ein BPTR auf die Adresse der *FileHandle*-Datenstruktur des aktuellen Eingabegeräts.

- ***cli_Command*ndFile 0x0024 (36)** Ein BSTR auf den Namen der aktuellen *EXECUTE*-Datei.
- **cli_Interactive 0x0028 (40)** Ein Boolescher Ausdruck, der wahr ist, wenn das CLI interaktiv ist (Eingaben über das CLI-Fenster möglich sind).
- **cli_Background 0x002B (44)** Ein Boolescher Ausdruck, der wahr ist, wenn das CLI mit dem Amiga-DOS-Befehl *RUN* gestartet wurde.
- **cli_CurrentOutput 0x0030 (48)** Ein BPTR auf die *FileHandle*-Datenstruktur des voreingestellten Eingabegeräts (In Modula-2 ein *FileHandlePtr*).
- **cli_DefaultStack 0x0034 (52)** Die Größe des beim Starten des CLI-Prozesses voreingestellten Stacks in Worten.
- **cli_StandardOutput 0x0038 (56)** Ein BPTR auf die *FileHandle*-Datenstruktur des voreingestellten Ausgabegeräts (In Modula-2 ein *FileHandlePtr*).
- **cli_Module 0x002B (60)** Ein BPTR auf die Segmentliste des aktuellen CLI-Befehls. Dort befindet sich im Speicher der ausführbare Code des Befehls.

6

DateStamp

Offset		C (libraries/dos.h)	Modula-2 (Dos)
Hex	Dez	struct DateStamp	Date
		{	RECORD
0000	0	LONG ds_Days;	days:LONGINT;
0004	4	LONG ds_Minute;	minute:LONGINT;
0008	8	LONG ds_Tick;	tick:LONGINT;
000C	12	};	END;

Datenstrukturenreferenz »DOS«: *DeviceList, FileInfoBlock, RootNode.*

Routinenreferenz »DOS«: *DateStamp.*

Beschreibung

In dieser Datenstruktur wird ein Datum gespeichert. Es liegt vor als die Anzahl von Tagen, Minuten und 1/50 Sekunden, die seit dem 01.01.1978 um 00:00:00 (Uhr) vergangen sind. *DateStamp* ist in der *FileInfoBlock*-Datenstruktur enthalten und beinhaltet das Erstellungsdatum und die Erstellungsuhrzeit einer Datei.

- **ds_Days0x0000 (0)** Die Anzahl der Tage seit dem 1. Januar 1978.
- **ds_Minute 0x0004 (4)** Die Tagesuhrzeit in Minuten.
- **ds_Tick 0x0008 (8)** Die 50stel Sekunden.

DeviceList

Offset		C (libraries/dos.h)	Modula-2 (Dos)
Hex	Dez	struct DeviceList	DeviceList =
		{	RECORD
0000	0	BPTR dl_Next;	next:DeviceListPtr;
			pad1,pad2,pad3:BYTE;
0004	4	LONG dl_Type;	type:DeviceListType;
0008	8	struct MsgPort * dl_Task;	task:ProcessId;
000C	12	BPTR dl_Lock;	lock:FileLockPtr;
			CASE :DeviceListType OF
			| device,directory:
			handler:BSTR;
			stackSize:LONGINT;
			priority:LONGINT;
			startup:LONGINT;
			segList:BPTR;
			globVec:BPTR;
			| volume:
0010	16	struct DateStamp dl_VolumeDate;	volumeDate:Date;
001C	28	BPTR dl_LockList;	lockList:FileLockPtr;
0020	32	LONG dl_DiskType;	diskType:LONGINT;
0024	36	LONG dl_unused;	unused:LONGINT
			END;
0028	40	BSTR * dl_Name;	name:BSTR;
002C	44	};	END;

Assembler — Syntax: Strukturname = *DevList*

Datenstrukturenreferenz — »DOS«: *DeviceList*, *DosInfo*, *InfoData*.

Routinenreferenz — Keine.

Beschreibung

Diese Datenstruktur beschreibt ein Volume (z.B. Diskette oder Festplatte). Je nach Wert von *dl_Type* werden die Felder *dl_VolumeDate* bis *dl_unsused* verschieden interpretiert. In »C« wird dann, wenn *dl_Type* den Wert *DLT_DEVICE* bzw. *DLT_DIRECTORY* hat, die Datenstruktur zu der unten beschriebenen *DeviceNode*-Datenstruktur. Sie beschreibt dann ein Device bzw. ein Directory. Da beide Datenstrukturen baugleich sind, ist die Modula-2-Version als varianter *RECORD* definiert. Alle Devices, Verzeichnisse und Volumes werden in einer über *dl_Next*-verketteten Liste gespeichert.

❑ **dl_Next 0x0000 (0)** — Ein BPTR auf die nächste *DeviceList*-Datenstruktur (bzw. *DeviceNode*-Datenstruktur) in der Liste.

❑ **dl_Type 0x0004 (4)**

Gibt an, ob die Datenstruktur ein Device oder ein Directory beschreibt. Mögliche Werte sind (in Modula-2 sind diese Werte als Aufzählungstyp *DeviceListType* definiert):

C	Modula-2	Hex	Beschreibung
DLT_DEVICE	device	0000	Die Datenstruktur beschreibt ein Device. In C und Assembler ist es also eine DeviceNode-Datenstruktur.
DLT_DIRECTORY	directory	0001	Die Datenstruktur bechreibt ein Directory. In C und Assembler ist es also eine DeviceNode-Datenstruktur.
DLT_VOLUME	volume	0002	Die Datenstruktur beschreibt ein Volume.

❑ **dl_Task 0x0008 (8)**

Falls das Volume im Laufwerk ist, steht hier die Prozeßkennung des Handlerprozesses (ein BPTR auf das *MsgPort*-Feld der *Process*-Datenstruktur), sonst *NULL*.

6

❑ **dl_Lock 0x000C (12)**

Bei residenten Devices steht hier *NULL*, sonst ein BPTR-Adreßzeiger auf die *FileLock*-Datenstruktur des Root-Verzeichnisses des Filesystems.

❑ **dl_VolumeDate 0x0010 (16)**

Eine eingebundene *DateStamp*-Datenstruktur, die das Erstellungsdatum des Volumes beinhaltet und zur Unterscheidung von Volumes mit gleichen Namen dient.

❑ **dl_LockList 0x001C (28)**

Hier wird die Liste der aktiven »FileLocks« des Volumes gespeichert, falls das Volume entfernt wird (Diskette aus dem Laufwerk genommen wird).

❑ **dl_DiskType 0x0020 (32)**

Dieser Parameter gibt an, um welchen Diskettentyp es sich handelt. Da es sich immer um eine Amiga-DOS-Diskette handeln muß, steht hier auch immer »DOS\n«.

❑ **dl_unsused 0x0024 (36)**

Ein nur quasi unbenutzter Parameter, denn dieses Feld wird benötigt, wenn dies eine *DeviceNode*-Datenstruktur ist.

❑ **dl_Name 0x0028 (40)**

Ein BSTR auf den Namen des Volumes.

DeviceNode

Offset		C (libraries/dos.h)	Modula-2 (Dos)
Hex	Dez	struct DeviceNode	DeviceList
		{	RECORD
0000	0	BPTR dn_Next;	next:DeviceListPtr;
			pad1,pad2,pad3:BYTE;
0004	4	LONG dn_Type;	type:DeviceListType;
0008	8	struct MsgPort * dl_Task;	task:ProcessId;
000C	12	BPTR dn_Lock;	lock:FileLockPtr;
			CASE :DeviceListType OF
			| device,directory:
0010	16	BSTR *dn_Handler;	handler:BSTR;
0014	20	LONG dn_StackSize;	stackSize:LONGINT;
0018	24	LONG dn_Priority;	priority:LONGINT;
001C	28	BPTR dn_Startup;	startup:FileSysStartupMsgPtr;
0020	32	BPTR dn_SegList ;	segList:BPTR;
0024	36	BPTR dn_ GlobVec	globVec:BPTR;
			| volume:
			volumeDate:Date;
			lockList:FileLockPtr;
			diskType:LONGINT;
			unused:LONGINT
			END;
0028	40	BSTR * dn_Name	name:BSTR;
002C	44	};	END;

Datenstrukturenreferenz »Dos«: *DeviceList, DosInfo, InfoData.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur beschreibt ein Device, ein Volume oder ein Directory. Je nach Wert von *dl_Type* werden die Felder *dl_VolumeDate* bis *dl_unsused* verschieden interpretiert. In C und Assembler wird dann, wenn *dl_Type* den Wert *DLT_VOLUME* hat, die Datenstruktur zu der oben beschriebenen *DeviceNode*-Datenstruktur. Da beide Datenstrukturen baugleich sind, ist die Modula-2-Version als varianter *RECORD* definiert. Alle Devices, Verzeichnisse und Volumes werden in einer über *dl_Next*-verketteten Liste gespeichert.

❑ **dn_Next 0x0000 (0)** Ein BPTR auf die nächste *DeviceNode*-Datenstruktur (bzw. *DeviceList*) in der Liste.

C	Modula-2	Dez	Beschreibung
DLT_DEVICE	device	0	Die Datenstruktur beschreibt ein Device.
DLT_DIRECTORY	directory	1	Die Datenstruktur beschreibt ein Directory.
DLT_VOLUME	volume	2	Die Datenstruktur beschreibt ein Volume. In C und Assembler ist es also eine DeviceNode-Datenstruktur.

- **dn_Type 0x0004 (4)** Die Bits dieser 2-Byte-Marke geben an, ob die Datenstruktur ein Device oder ein Directory beschreibt (in Modula-2 sind diese Werte als Aufzählungstyp *DeviceListType* definiert):

- **dn_Task 0x0008 (8)** Bei einem residenten Device steht hier die Prozeßkennung des Handlerprozesses (BPTR auf das *MsgPort*-Feld der *Process*-Datenstruktur), ansonsten *NULL*.

6

- **dn_Lock 0x000C (12)** Bei residenten Devices steht hier *NULL*, sonst ein BPTR auf die *FileLock*-Datenstruktur des Root-Verzeichnisses des Filesystems.

- **dn_Handler 0x0010 (16)** Falls der »Handler« nicht im Speicher ist, muß in diesem Feld ein BSTR-Adreßzeiger stehen. Dieser zeigt auf den Namen (incl. Pfadangabe) des »Handlers« auf der Diskette. Dieser kann dann mittels der *LoadSeg*-Routine geladen werden.

- **dn_StackSize 0x0014 (20)** Ist der »Handler« geladen, so steht hier die Größe des zugehörigen Stacks.

- **dn_Priority 0x0018 (24)** Die Priorität des »Handlers«.

- **dn_Startup 0x001C (28)** Ein Adreßzeiger auf eine »FileSystemStartupMessage«, die dem Handlerprozeß beim Start übergeben wird.

- **dn_SegList 0x0020 (32)** Ein BPTR-Adreßzeiger auf die Segmentliste des »Handlers«. Er ist *NULL*, falls der »Handler« nicht im Speicher ist.

- **dn_GlobVec 0x0024 (36)** Ein BPTR auf eine private Sprungtabelle des Handlerprozesses.

- **dn_Name 0x0028 (40)** Ein BSTR auf den Namen des Device bzw. des Verzeichnisses.

DosEnvec

Offset		C (libraries/filehandler.h)	Modula-2 (Dos)
Hex	Dez	struct InfoData	DosEnvec =
		{	RECORD
0000	0	ULONG de_TableSize;	tableSize:LONGCARD;
0004	4	ULONG de_SizeBlock;	sizeBlock:LONGCARD;
0008	8	ULONG de_SecOrg;	secOrg:LONGCARD;
000C	12	ULONG de_Surfaces;	surfaces:LONGCARD;
0010	16	ULONG de_SectorPerBlock;	sectorPerBlock:LONGCARD;
0014	20	ULONG de_BlocksPerTrack;	blocksPerTrack:LONGCARD;
0018	24	ULONG de_Reserved;	reserved:LONGCARD;
001C	28	ULONG de_PreAlloc;	preAlloc:LONGCARD;
0020	32	ULONG de_Interleave;	interleave:LONGCARD;
0024	36	ULONG de_LowCyl;	lowCyl:LONGCARD;
0028	40	ULONG de_HighCyl;	highCyl:LONGCARD;
002C	44	ULONG de_NumBuffers;	numBuffers:LONGCARD;
0030	48	ULONG de_BuffMemType:	buffMemType:LONGCARD;
0034	52	ULONG de_MaxTransfer;	maxTransfers:LONGCARD;
0038	56	ULONG de_Mask;	mask:LONGSET; LONGCARD;
003C	60	LONG de_BootPri;	bootPri:LONGINT; LONGCARD;
0040	64	ULONG de_DosType;	dosType:ARRAY [0..3] OF CHAR;
0044	68	}	END;

Datenstrukturenreferenz »Dos«: *FileSystemStartupMsg.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur gibt dem Devicehandler Auskunft über ein beliebiges Device. Ihre Adresse muß in dem *fssm_Environ*-Feld der *FileSystemStartupMsg*-Datenstruktur eingetragen werden. Wie Sie sehen, entsprechen die Felder dieser Datenstruktur den Einträgen der Mountlist. Der CLI-Befehl *MOUNT* erzeugt auch aus dem Mountlisteintrag eine solche Datenstruktur. Da die Mountlist je nach Device verschiedene Formen annehmen kann, ist die *DosEnvec*-Datenstruktur eigentlich ein Array von *ULONG*-Werten, die erst vom Handler ausgewertet werden. Die hier abgedruckte Version wird für alle Devices benötigt, die das *Trackdisk*-Device emulieren und als Handler das Filesystem (bzw. FastFileSystem) benutzen (wie beispielsweise ein Harddisktreiber).

- **de_TableSize 0x0000 (0)** Hier steht die Größe der gesamten Datenstruktur. Dieses Feld ist immer dann wichtig, wenn diese Datenstruktur als Array behandelt wird, weil sie dann von der hier beschriebenen Form abweicht.

- **de_SizeBlock 0x0004 (4)** Die Größe eines Datenblocks in Langwörtern (normalerweise 128).

- **de_SecOrg 0x0008 (8)** In der aktuellen Version unbenutzt. Sollte trotzdem immer auf 0 gesetzt werden.
- **de_Surfaces 0x000C (12)** Die Anzahl der Schreib-Lese-Köpfe des Gerätes (beim Diskettenlaufwerk 2).
- **de_SectorPerBlock 0x0010 (16)** In der aktuellen Version unbenutzt. Sollte immer auf 0 gesetzt werden.
- **de_BlocksPerTrack 0x0014 (20)** Die Anzahl der Blöcke pro Track. Beim normalen Diskettenlaufwerk sind es 11, bei Festplatten 17 (MFM) oder 26 (RLL).
- **de_Reserved 0x0018 (24)** Die Anzahl der für DOS reservierten Blöcke am Anfang der Partition (normalerweise 2).
- **de_PreAlloc 0x001C (28)** Die Anzahl der für DOS reservierten Blöcke am Ende der Partition.
- **de_Interleave 0x0020 (32)** Der Sektorversatz (Interleave). Für Diskettenlaufwerke 0, für Festplatten zwischen 0 und 26.
- **de_LowCyl 0x0024 (36)** Die Nummer des ersten Tracks der Partition (bei Diskettenlaufwerken, immer 0).
- **de_HighCyl 0x0028 (40)** Die Nummer des letzten Tracks der Partition (bei Diskettenlaufwerken 39 oder 79).
- **de_NumBuffers 0x002C (44)** Die Anzahl von 512-Byte langen Puffern, die für das Device angelegt werden sollen. Mehr Puffer bedeuten normalerweise einen schnelleren Zugriff.
- **de_BuffMemType 0x0030 (48)** Spezielle Anforderungen an den Pufferspeicher. Wie in der Mountlist gilt hier 3 = Chip-RAM, 1 = Fast-RAM
- **de_MaxTransfer 0x0034 (52)** Die maximale Anzahl der Bytes, die auf einmal übergeben werden können (normalerweise 13).
- **de_Mask 0x0038 (56)** Laut [KRA] eine Maske, um bestimmte Speicherbereiche auszublenden (Klingt gut, oder?).
- **de_BootPri 0x003C (60)** Die Priorität für »bootende« Devices (15 für Autoboot).
- **de_DosType 0x0040 (64)** Eine Zeichenkette, die die Version des Filesystems angibt. Dabei bedeuten als Langwort codiert 0x444F5300 das alte Filesystem, 0x444f5301 das neue Fast-Filesystem.

DosInfo

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez	struct DosInfo {	DosInfo = RECORD
0000	0	BPTR di_McName;	mcName:BSTR;
0004	4	BPTR di_DevInfo;	devInfo:DeviceListPtr;
0008	8	BPTR di_Devices;	devices:BPTR;
000C	12	BPTR di_Handlers;	handlers:BPTR;
0010	16	APTR di_NetHand;	netHand:ResidentSegmentPtr;
0014	20	};	END;

Datenstrukturenreferenz »Dos«: *RootNode*.

Routinenreferenz Keine.

6

Beschreibung

Diese Datenstruktur ist für Erweiterungen vorgesehen, die den Amiga netzwerkfähig machen.

❑ **di_McName 0x0000 (0)** Ein BPTR-Adreßzeiger auf den Namen dieses Computers im Netzwerk.

❑ **di_DevInfo 0x0004 (4)** Ein BPTR-Adreßzeiger auf die Liste der Devices.

❑ **di_Devices 0x0008 (8)** Dieser BPTR-Adreßzeiger ist immer *NULL*.

❑ **di_Handlers 0x000C (12)** Dieser BPTR-Adreßzeiger ist immer *NULL*.

❑ **di_NetHand 0x0010 (16)** Dieser Adreßzeiger ist vorgesehen für die Prozeßkennung des Netzwerkhandlers.

DosLibrary

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez	struct DosLibrary {	DosLibrary = RECORD
0000	0	struct Library dl_lib;	lib:Library;
0022	34	APTR dl_Root;	root:RootNodePtr;
0026	38	APTR dl_GV;	gv:ADDRESS;
002A	42	LONG dl_A2;	a2:LONGINT;
002E	46	LONG dl_A5;	a5:LONGINT;
0032	50	LONG dl_A6;	a6:LONGINT;
0036	54	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Dies ist eine globale Datenstruktur, deren Adresse von der Routine *OpenLibrary* zurückgegeben wird, wenn Sie die Dos-Library öffnen.

- **dl_Library 0x0000 (0)** Eine eingebundene *Library*-Datenstruktur wie im Kapitel über »Exec« behandelt. Sie beschreibt die Dos-Library und bindet sie in das System ein.
- **dl_Root 0x0022 (34)** Ein Adreßzeiger auf die *RootNode*-Datenstruktur, die weitere wichtige globale Informationen enthält.
- **dl_GV 0x0026 (38)** Ein Adreßzeiger auf eine globale Sprungtabelle von Amiga-DOS.
- **dl_A2 0x002A (42)** Ein systeminternes Register zur Zwischenspeicherung des gleichnamigen Prozessorregisters.
- **dl_A5 0x002E (46)** Ein systeminternes Register zur Zwischenspeicherung des gleichnamigen Prozessorregisters.
- **dl_A6 0x0032 (50)** Ein systeminternes Register zur Zwischenspeicherung des gleichnamigen Prozessorregisters.

6

DosPacket

Offset Hex	Offset Dez	C (libraries/dosextens.h)	Modula-2 (Dos)
		struct DosPacket	DosPacket =
		{	RECORD
0000	0	struct Message *dp_Link;	link:MessagePtr;
0004	4	struct MsgPort *dp_Port;	port:MsgPortPtr;
0008	8	LONG dp_Type;	type:LONGINT;
000C	12	LONG dp_Res1;	res1:LONGINT;
0010	16	LONG dp_Res2;	res2:LONGINT;
0014	20	LONG dp_Arg1;	arg1:LONGINT;
0018	24	LONG dp_Arg2;	arg2:LONGINT;
001C	28	LONG dp_Arg3;	arg3:LONGINT;
0020	32	LONG dp_Arg4;	arg4:LONGINT;
0024	36	LONG dp_Arg5;	arg5:LONGINT;
0028	40	LONG dp_Arg6;	arg6:LONGINT;
002C	44	LONG dp_Arg7;	arg7:LONGINT;
0030	48	};	END;

Datenstrukturenreferenz »Dos«: *StandardPacket.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur dient zur Kommunikation zwischen verschiedenen Prozessen. Die Adresse einer solchen Datenstruktur wird im *Name*-Feld einer *Message*-Datenstruktur gespeichert. Solche »Packets« werden von DOS zur Übermittlung von Befehlen an Device-Handler (z.B. Laufwerkshandler) verwendet. Die Datenstruktur muß an einer durch 4 teilbaren Adresse liegen.

- **dp_Link 0x0000 (0)** Ein Adreßzeiger zurück auf die *Message*-Datenstruktur, die benutzt wird, um dieses »Packet« zu senden.

- **dp_Port 0x0004 (4)** Ein Adreßzeiger auf eine *MsgPort*-Datenstruktur, zu der die Antwort von dem Empfänger geschickt werden soll. Dies soll normalerweise nicht die *MsgPort*-Datenstruktur sein, die als ID des Prozesses dient (also in der *Process*-Datenstruktur enthalten ist).

6

- **dp_Type 0x0008 (8)** In diesem Feld steht der Typ dieses »Packets«. Bei Nachrichten, die an den DOS-Handler geschickt werden, also der Befehl, der ausgeführt werden soll. Der Inhalt dieses Feldes bestimmt die Bedeutung der nachfolgenden. Die verfügbaren Befehle und die zugehörige Bedeutung der restlichen Felder können Sie der Auflistung im Abschnitt (»Die DOS-Packets«) entnehmen.

- **dp_Res1 0x000C (12)** In diesem Feld gibt der Empfänger der Nachricht bei der Antwort Daten zurück. Bei dem »File-System« Aufruf *Write* wird beispielsweise die Länge der geschriebenen Daten zurückgegeben.

- **dp_Res2 0x0010 (16)** In diesem Feld gibt der Empfänger der Nachricht bei der Antwort Daten zurück. Bei »File-System« Aufrufen beispielsweise die Antwort der *IoErr*-Routine.

- **dp_Arg1 0x0014 (20)** Das erste Argument, das dem Empfänger als Information übergeben wird (werden kann).

- **dp_Arg2 0x0018 (24)** Das zweite Argument, das dem Empfänger als Information übergeben wird (werden kann).

- **dp_Arg3 0x001C (28)** Das dritte Argument, das dem Empfänger als Information übergeben wird (werden kann).

- **dp_Arg4 0x0020 (32)** Das vierte Argument, das dem Empfänger als Information übergeben wird (werden kann).

- **dp_Arg5 0x0024 (36)** Das fünfte Argument, das dem Empfänger als Information übergeben wird (werden kann).

❑ **dp_Arg6 0x0028 (40)** Das sechste Argument, das dem Empfänger als Information übergeben wird (werden kann).

❑ **dp_Arg7 0x002C (44)** Das siebte Argument, das dem Empfänger als Information übergeben wird (werden kann).

FileHandle

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	**Dez**	struct FileHandle	FileHandle =
		{	RECORD
0000	0	struct Message *fh_Link;	link:MessagePtr;
0004	4	struct MsgPort *fh_Port;	port:MsgPortPtr;
0008	8	struct MsgPort *fh_Type;	type:ProcessId;
000C	12	LONG fh_Buf;	buf:LONGINT;
0010	16	LONG fh_Pos;	pos:LONGINT;
0014	20	LONG fh_End;	end:LONGINT;
0018	24	LONG fh_Funcs;	func1:LONGINT;
001C	28	LONG fh_Func2;	func2:LONGINT;
0020	32	LONG fh_Func3;	func3:LONGINT;
0024	36	LONG fh_Args;	arg1:LONGINT;
0028	40	LONG fh_Arg2;	arg2:LONGINT;
002C	44	};	END;

6

Datenstrukturenreferenz »Dos«: *CommandLineInterface*, *Process*.

Routinenreferenz »Dos«: *Close*, *Execute*, *Input*, *IsInteractive*, *Open*, *Output*, *Read*, *Seek*, *WaitForChar*, *Write*.

⇨ Beschreibung

Die Datenstruktur identifiziert eine Datei. Wenn Sie eine Datei mit Hilfe der *Open*-Routine öffnen, bekommen Sie einen Adreßzeiger auf die zugehörige *FileHandle*-Datenstruktur zurück. Sie brauchen diesen, um diese Datei bearbeiten zu können.

❑ **fh_Link 0x0000 (0)** Nicht benutzt.

❑ **fh_Port 0x0004 (4)** Dieser Wert ist null, falls die Datei interaktiv ist.

❑ **fh_Type 0x0008 (8)** Prozeßkennung des Handlerprozesses (ein Adreßzeiger auf das *MsgPort*-Feld der entsprechenden *Process*-Datenstruktur). Dort werden die DOS-Packets abgeschickt.

❑ **fh_Buf 0x000C (12)** Ein Adreßzeiger auf einen internen Puffer, der bei der Arbeit mit der Datei verwendet wird.

- **fh_Pos 0x0010 (16)** Die aktuelle Zeichenposition innerhalb des Puffers.
- **fh_End 0x0014 (20)** Die Endposition innerhalb des Puffers.
- **fh_Funcs 0x0018 (24)** Die Adresse der Funktion, die zum Auslesen des Puffers benutzt wird.
- **fh_Func2 0x001c (28)** Die Adresse der Funktion, die zum Beschreiben des Puffers verwendet wird.
- **fh_Func3 0x0020 (32)** Die Adresse der Funktion, die zum Schließen der Datei verwendet wird.
- **fh_Args 0x0024 (36)** Ein Argument zur internen Verwendung. Seine genaue Funktion hängt von der Art des »FileHandlers« ab.
- **fh_Arg2 0x0028 (40)** Ein Argument zur internen Verwendung. Seine genaue Funktion hängt von der Art des »FileHandlers« ab.

6

FileInfoBlock

Offset		C (libraries/dos.h)	Modula-2 (Dos)
Hex	**Dez**	struct FileInfoBlock	FileInfoBlock =
		{	RECORD
0000	0	LONG fib_DiskKey;	diskKey:LONGINT;
0004	4	LONG fib_DirEntryType;	dirEntryType:LONGINT;
0008	8	char fib_FileName[108];	fileName:ARRAY[0..107] OF CHAR;
0074	116	LONG fib_Protection;	protection:ProtectionFlagSet;
0078	120	LONG fib_EntryType;	entryType:LONGINT;
007C	124	LONG fib_Size;	size:LONGINT;
0080	128	LONG fib_NumBlocks;	numBlocks:LONGINT;
0084	132	struct DateStamp fib_Date;	date:Date;
0090	144	char fib_Comment[80];	comment:ARRAY [0..79] OF CHAR;
00E0	224	char fib_Reserved[36];	reserved:ARRAY [0..35] OF CHAR;
0104	260	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz »DOS«: *Examine*, *ExNext*, *Info*.

Beschreibung

In dieser Datenstruktur speichert »Dos« die wichtigsten Informationen über eine Datei. Falls Sie eine Datei untersuchen wollen, können Sie sich die zugehörige *FileInfoBlock*-Datenstruktur mit Hilfe des *Examine*- bzw. *ExNext*-Befehls besorgen. Dieser Datensatz enthält im wesentlichen die gleichen Daten wie der *FileHeader*-Block eines Files auf Diskette. Wird eine *FileInfoBlock*-Datenstruktur an DOS-Routinen übergeben, dann muß diese an einer durch 4 teilbaren Adresse liegen. Dazu allokieren Sie sie am besten mit der *AllocMem*-Routine aus »Exec«.

- **fib_DiskKey 0x0000 (0)** Die Nummer des ersten Blocks des Eintrags auf Diskette.
- **fib_DirEntryType 0x0004 (4)** Ist dieses Feld größer als null, dann handelt es sich bei dem Eintrag um ein Verzeichnis, sonst um eine Datei.
- **fib_FileName 0x0008 (8)** Eine eingebundene Zeichenkette, die den Namen des Eintrags beinhaltet.
- **fib_Protection 0x0074 (116)** Die »Protection-Bits« des Eintrags.
- **fib_EntryType 0x0078 (120)** Eine Systeminterne Kennung.
- **fib_Size 0x007C (124)** Die Länge des Eintrags in Bytes.
- **fib_NumBlocks 0x0080 (128)** Die Anzahl der Blocks, die der Eintrag belegt.
- **fib_Date 0x0084 (132)** Eine *DateStamp*-Datenstruktur, die das Erstellungsdatum des Eintrags beinhaltet.
- **fib_Comment 0x0090 (144)** Der zu dem Eintrag gehörende Kommentar als Zeichenkette.
- **fib_Padding 0x00E0 (224)** Ein Zwischenspeicher für diverse Systemzwecke.

FileLock

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	**Dez**	struct FileLock	FileLock =
		{	RECORD
0000	0	BPTR fl_Link;	link:FileLockPtr;
0004	4	LONG fl_Key;	key:LONGINT;
0008	8	LONG fl_Access;	access:LONGINT;
000C	12	struct MsgPort * fl_Task;	task:ProcessId;
0010	16	BPTR fl_Volume;	volume:DeviceListPtr;
0014	20	};	END;

Datenstrukturenreferenz »DOS«: *DeviceList.*

Routinenreferenz »DOS«: *CreateDir*, *CurrentDir*, *DupLock*, *Examine*, *ExNext*, *Info*, *Lock*, *ParentDir*, *Unlock*.

Beschreibung

Diese Datenstruktur dient zur Identifizierung eines Eintrags innerhalb des »Filesystems«. Sie wird benötigt, um die Zugriffe auf ein File zu koordinieren. Anders als die *FileHandle*-Datenstruktur dient diese Datenstruktur zur Koordination der Zugriffe verschiedener Tasks auf eine Datei und nicht deren Verwaltung. Daher kann eine *FileLock*-Datenstruktur auch für ein Verzeichnis geholt werden. Ein Adreßzeiger auf eine *FileLock*-Datenstruktur muß an fast alle DOS-Routinen übergeben werden, die auf Einträge im »Filesystem« zugreifen. Es können gleichzeitig mehrere »Locks« auf die gleiche Datei von verschiedenen Prozessen angefordert werden. Dabei muß es sich allerdings um »FileLocks« des Typs *SHARED_LOCK (ACCESS_READ)* handeln.

❑ **fl_Link 0x0000 (0)** Mit Hilfe dieses Feldes verkettet Amiga-DOS die »Locks«.

❑ **fl_Key 0x0004 (4)** Hier steht die Nummer des ersten Blocks (FileHeader) oder des Verzeichnisses auf Diskette.

❑ **fl_Access 0x0008 (8)** Gibt die Art des Zugriffs an, der über dieses »Lock« zugelassen ist.

C	Modula-2	Dez	Beschreibung
SHARED_LOCK	sharedLock	72	Nur Lesezugriff ist möglich.
ACCESS_READ	accessRead	72	Nur Lesezugriff ist möglich.
EXCLUSIVE_LOCK	exclusiveLock	71	Nur Schreibzugriff ist möglich.
ACCESS_WRITE	accessWrite	71	Nur Schreibzugriff ist möglich.

❑ **fl_Task 0x000C (12)** Ein Adreßzeiger auf die *MsgPort*-Datenstruktur des Handlerprozesses des Device, zu dem dieses File gehört.

6

❑ **fl_Volume 0x0010 (16)** Ein BPTR-Adreßzeiger auf eine *DeviceList*-Datenstruktur, die nähere Informationen über das Device enthält, zu dem das File gehört.

FileSystemStartupMsg

Offset		C (libraries/filehandler.h)	Modula-2 (DOS)
Hex	**Dez**	struct InfoData	FileSysStartupMsg =
		{	RECORD
0000	0	ULONG fssm_Unit;	unit:LONGCARD;
0004	4	BSTR fssm_Device;	device:BSTR;
0008	8	BPTR fssm_Environ;	environ: DosEnvecPtr;
000C	12	ULONG fssm_Flags	flags:LONGSET;
0010	16	};	END;

Datenstrukturenreferenz »Dos«: *DeviceNode.*

Routinenreferenz Keine.

Beschreibung

Bei Devices, die neu geöffnet werden, steht in dem *dn_Startup*-Feld, der *DeviceNode*-Datenstruktur, ein BPTR auf eine *FileSystemStartupMsg*-Datenstruktur. Sie sagt dem System, was bei der Initialisierung des Devices (mit der *OpenDevice*-Routine) zu tun ist und wie das Device physikalisch aufgebaut ist.

❑ **fssm_Unit 0x0000 (0)** Hier steht die Exec-Unitnummer (z.B. die Laufwerksnummer).

- ❑ **fssm_Device 0x0004 (4)** — Ein BSTR-Adreßzeiger auf eine nullterminierte Zeichenkette, die den Namen des Devices enthält. Sie wird mit dem bei *OpenDevice* angegebenen Namen verglichen.
- ❑ **fssm_Environ 0x0008 (8)** — Ein Adreßzeiger auf eine *DosEnvec*-Datenstruktur oder ein beliebiges *ULONG*-Array, in dem weitere Informationen zum Device enthalten sind.
- ❑ **fssm_Flags 0x000C (12)** — Einige Flags, die für die Routine *OpenDevice* benötigt werden.

InfoData

Offset		C (libraries/dos.h)	Modula-2 (Dos)
Hex	**Dez**	struct InfoData	InfoData =
		{	RECORD
0000	0	LONG id_NumSoftErrors;	numSoftErrors:LONGINT;
0004	4	LONG id_UnitNumber;	unitNumber:LONGINT;
0008	8	LONG id_DiskState;	diskState:LONGINT;
000C	12	LONG id_NumBlocks;	numBlocks:LONGINT;
0010	16	LONG id_NumBlocksUsed;	numBlocksUsed:LONGINT;
0014	20	LONG id_BytesPerBlock;	bytesPerBlock:LONGINT;
0018	24	LONG id_DiskType;	diskType:LONGINT;
001C	28	BPTR id_VolumeNode;	volumeNode:DeviceListPtr;
0020	32	LONG id_InUse;	inUse:LONGINT;
0024	36	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz »DOS«: *Info.*

Beschreibung

In dieser Datenstruktur werden diverse Informationen über ein Volume (also z.B. eine Diskette oder eine »RAM DISK«) gespeichert. Mit Hilfe der *Info*-Routine können Sie eine solche Datenstruktur mit den Daten eines Volumes initialisieren lassen. Zu beachten ist, daß die *InfoData*-Datenstruktur an einer durch vier teilbaren Adresse liegen muß.

- ❑ **id_NumSoftErrors 0x0000 (0)** — Falls es sich bei dem Volume um eine Diskette/Festplatte handelt, finden Sie hier die Anzahl der fehlerhaften Blocks/Tracks.
- ❑ **id_UnitNumber 0x0004 (4)** — Die Nummer des Laufwerks (–1 für die »RAM DISK«).
- ❑ **id_DiskState 0x0008 (8)** — Momentaner Zustand der Diskette. Hier können folgende Werte stehen:

6

C	Modula-2	Dez	Beschreibung
ID_WRITE_PROTECTED	writeProtected	80	Die Diskette ist schreibgeschützt.
ID_VALIDATING	validating	81	Die Diskette wird überprüft.
ID_VALIDATED	validated	82	Die Diskette ist OK.

- **id_NumBlocks 0x000C (12)** Die Anzahl der Blocks, die das Volume hat.
- **id_NumBlocksUsed 0x0010 (16)** Anzahl der bereits belegten Blocks des Volumes.
- **id_BytesPerBlock 0x0014 (20)** Die Größe eines einzelnen Blocks in Bytes.
- **id_DiskType 0x0018 (24)** Hier steht Näheres über eine eingelegte Diskette. Folgende Werte sind möglich (in Modula-2 werden die Zeichenketten mit Hilfe von *CAST* in *LONGINT*-Werte umgewandelt).

C	Modula-2	Wert	Beschreibung
ID_NO_DISK_PRESENT	noDiskPresent	71L	Es ist keine Diskette im Laufwerk.
ID_UNREADABLE_DISK	unreadableDisk	'BAD\0'	Die Diskette ist nicht lesbar.
ID_DOS_DISK	dosDisk	'DOS\0'	Die Diskette ist OK (Amiga-DOS).
ID_NOT_REALLY_DOS	notReallyDos	'NDOS'	Es ist keine Amiga-DOS-Diskette.
ID_KICKSTART_DISK	kickstartDisk	'KICK'	Es ist eine Kickstart-Diskette.

- **id_VolumeNode 0x001C (28)** Ein BPTR auf die zugehörige *DeviceList*-Datenstruktur.
- **id_InUse 0x0020 (32)** Falls auf die Datei gerade zugegriffen wird, steht hier ein Wert ungleich null.

Process

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez	struct Process	Process =
		{	RECORD
0000	0	struct Task pr_Task;	task:Task;
005C	92	struct MsgPort pr_MsgPort;	msgPort:MsgPort;
007E	126	WORD pr_Pad;	pad:WORD;
0080	128	BPTR pr_SegList;	segList:BPTR;
0084	132	LONG pr_StackSize;	stackSize:LONGINT;
0088	136	APTR pr_GlobVec;	globVec:ADDRESS;
008C	140	LONG pr_TaskNum;	taskNum:LONGINT;

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez		
0090	144	BPTR pr_StackBase;	stackBase:BPTR;
0094	148	LONG pr_Result2;	result2:LONGINT;
0098	152	BPTR pr_CurrentDir;	currentDir:FileLockPtr;
009C	156	BPTR pr_CIS;	cis:FileHandlePtr;
00A0	160	BPTR pr_COS;	cos:FileHandlePtr;
00A4	164	APTR pr_ConsoleTask;	consoleTask:ProcessId;
00A8	168	APTR pr_FileSystemTask;	fileSystemTask:ProcessId;
00AC	172	BPTR pr_CLI;	cli:CommandLineInterfacePtr;
00B0	176	APTR pr_ReturnAddr;	returnAddr:ADDRESS;
00B4	180	APTR pr_PktWait;	pktWait:ADDRESS;
00B8	184	APTR pr_WindowPtr;	windowPtr:ADDRESS;
00BC	188	};	END;

6

Datenstrukturenreferenz Keine.

Routinenreferenz »Dos«: *CreateProcess.*
»Exec«: *FindTask.*

Beschreibung

Diese Datenstruktur identifiziert einen Prozeß. Sie wird bei der Erzeugung des Prozesses mit der *CreateProcess*-Routine initialisiert und enthält Informationen über die Art des Prozesses, die zugehörigen Handler, die zugehörigen Tasks etc. Sie ist eigentlich nur eine erweiterte *Task*-Datenstruktur. Der Unterschied zwischen einem einfachen und einem in einen Prozeß eingebundenen Task besteht darin, daß der zweite uneingeschränkt auf »Dos«-Routinen, Datenstrukturen und Handler zugreifen kann. Jedem Programm, das vom CLI oder von der Workbench gestartet wird, wird eine solche Datenstruktur zugeordnet. Sie können die Anfangsadresse einer solchen Datenstruktur mit der *FindTask*-Routine aus »Exec« ermitteln (für den eigenen Prozeß dabei *NULL* als Namen angeben). Alle Routinen und Datenstrukturen benutzen allerdings zur Identifikation eines Prozesses nicht die Anfangsadresse, sondern einen BPTR auf das *MsgPort*-Feld.

❑ **pr_Task 0x0000 (0)** Eine vollständig eingebundene *Task*-Datenstruktur wie im Kapitel über die Exec-Library beschrieben. Sie sorgt dafür, daß »Dos« Prozesse in das Multitasking des Amiga eingebunden werden. Ihre Adresse wird von der *FindTask*-Routine geliefert.

❑ **pr_MsgPort 0x005C (92)** Eine eingebundene *MsgPort*-Datenstruktur wie im Kapitel über die Exec-Library beschrieben. Dieser Port dient als Kennung des Prozesses. An ihn werden auch alle an diesen Prozeß gerichteten Nachrichten geschickt.

❑ **pr_Pad 0x007E (126)** Ein Füllbyte ohne weitere Bedeutung.

- **pr_SegList 0x0080 (128)** Ein BPTR auf die Segmentliste des Programms. Er zeigt auf ein Feld von Adreßzeigern, von denen der vierte auf das erste Segment zeigt. Die Segmente sind über BPTR-Adreßzeiger verkettet. Dabei gibt das erste Element die Größe des Feldes an.
- **pr_StackSize 0x0084 (132)** Die Größe des Stack-Prozesses in Bytes, die ihm beim Aufruf von der Routine *CreateProcess* zugewiesen wird. Die Größe des Prozeßstacks ist nicht mit der der CLI-Befehle identisch.
- **pr_GlobVec 0x0088 (136)** Ein Adreßzeiger auf die globale Sprungtabelle von AmigaDOS.
- **pr_TaskNum 0x008C (140)** Ist dies ein CLI-Prozeß, so steht hier die Nummer dieses Prozesses in *TaskTable* der *RootNode*-Datenstruktur, sonst null.
- **pr_StackBase 0x0090 (144)** Ein BPTR auf das Ende des Prozeßstacks.
- **pr_Result2 0x0094 (148)** Hier steht die Nummer der Fehlermeldung, die von der Routine *IoErr* geliefert wird.
- **pr_CurrentDir 0x0098 (152)** Ein BPTR auf die *FileLock*-Stuktur des aktuellen Verzeichnisses. Dieser wird von der Routine *CurrentDir* geliefert.
- **pr_CIS 0x009C (156)** Ein BPTR auf die *FileHandle*-Datenstruktur des CLI-Eingabestromes.
- **pr_COS 0x00A0 (160)** Ein BPTR auf die *FileHandle*-Datenstruktur des CLI-Ausgabestromes.
- **pr_ConsoleTask 0x00A4 (164)** Ein BPTR auf das *MsgPort*-Feld der *Process*-Datenstruktur des zugehörigen Console-Handlers des aktuellen Fensters.
- **pr_FileSystemTask 0x00A8 (168)** Ein BPTR auf das *MsgPort*-Feld der *Process*-Datenstruktur des zugehörigen Handlers des aktuellen Laufwerks.
- **pr_CLI 0x00AC (172)** Falls dies kein CLI-Prozeß ist, so steht hier *NULL*, sonst ein BPTR auf eine *CommandLineInterface*-Datenstruktur.
- **pr_ReturnAddr 0x00B0 (176)** Ein Adreßzeiger auf die auf dem Stack abgelegte Rücksprungadresse, die beim Verlassen des Programms mit der *EXIT*-Funktion angesprungen wird.
- **pr_PktWait 0x00B4 (180)** Ein Adreßzeiger auf eine Funktion, die aufgerufen wird, wenn der Prozeß anhält, um auf eine Nachricht zu warten. Normalerweise steht hier jedoch *NULL*.
- **pr_WindowPtr 0x00B8 (184)** Ein Adreßzeiger auf das Fenster, in dem die »Dos«-Requester mit Fehlermeldungen bzw. Anforderungen an den Benutzer erscheinen. Steht hier *NULL*, dann wird der Requester im Fenster der Workbench erzeugt. Wenn Sie wollen, daß die Fehlermeldung mittels *Write* ausgegeben wird, geben Sie hier –1 an.

RootNode

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez	struct RootNode	RootNode =
		{	RECORD
0000	0	BPTR rn_TaskArray;	taskArray:TaskArrayPtr;
0004	4	BPTR rn_ConsoleSegment;	consoleSegment:BPTR;
0008	8	struct DateStamp rn_Time;	time:Date;
0032	20	LONG rn_RestartSeg;	restartSeg:BPTR;
0036	24	BPTR rn_Info;	info:DosInfoPtr;
003A	28	BPTR rn_FileHandlerSegment;	fileHandlerSegment:BPTR;
0040	32	};	END;

Datenstrukturenreferenz »Dos«: *DosLibrary.*

Routinenreferenz Keine.

Beschreibung

Eine globale Datenstruktur für »Dos«, die wichtige Daten über den momentanen Zustand des Systems speichert.

❑ **rn_TaskArray 0x0000 (0)** Ein BPTR auf eine Tabelle, in der die BPTR auf die *MsgPort*-Datenstrukturen aller aktiven CLI-Prozesse gespeichert sind. Das nullte Feld beinhaltet die Länge der Tabelle.

❑ **rn_ConsoleSegment 0x0004 (4)** Ein BPTR auf die Segmentliste des CLI-Codes, der von *NEWCLI* zur Erzeugung eines neuen CLIs benutzt wird.

❑ **rn_Time 0x0008 (8)** Die aktuelle Zeit in Form einer eingebundenen *DateStamp*-Datenstruktur.

❑ **rn_RestartSeg 0x0032 (20)** Ein BPTR auf die Segmentliste des Codes, der immer dann aufgerufen wird, wenn eine neue Diskette ins Laufwerk eingelegt wird.

❑ **rn_Info 0x0036 (24)** Ein BPTR auf die globale *Info*-Datenstruktur.

❑ **rn_FileHandlerSegment 0x003A (28)** Ein Adreßzeiger auf die Segmentliste des Filehandlers.

StandardPacket

Offset		C (libraries/dosextens.h)	Modula-2 (Dos)
Hex	Dez	struct StandardPacket	StandardPacket =
		{	RECORD
0000	0	struct Message sp_Msg;	msg:Message;
0014	20	struct DosPacket sp_Pkt;	pkt:DosPacket;
0044	68	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

In dieser Datenstruktur sind eine *Message*- und eine *DosPacket*-Datenstruktur zusammengefaßt. Sie werden benötigt, um ein »Packet« zu senden.

sp_Msg 0x0000 (0) Eine *Message*-Datenstruktur wie im Kapitel über die Exec-Library beschrieben. Ihre Adresse wird zum Senden des Packets an die Routine *PutMsg* übergeben. Das Name-Feld muß dabei auf *sp_Pkt* zeigen.

sp_Pkt 0x0014 (20) Eine *DosPacket*-Datenstruktur, die den Befehl und die Argumente beinhaltet.

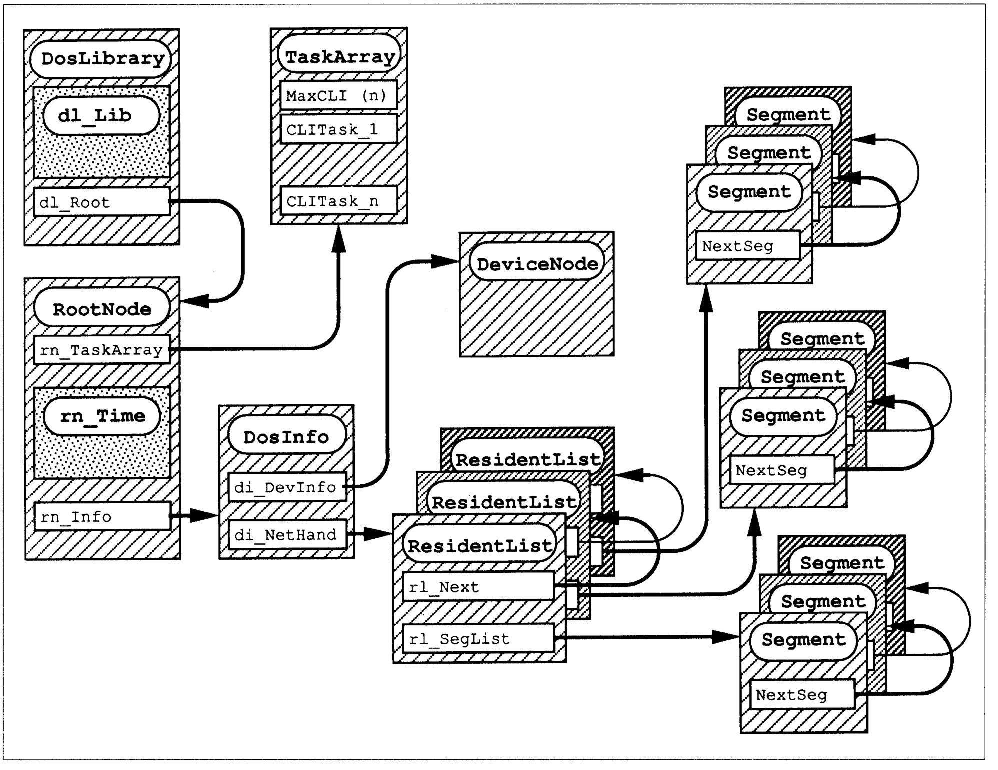


Abb. 6.1: Die DOS-Library

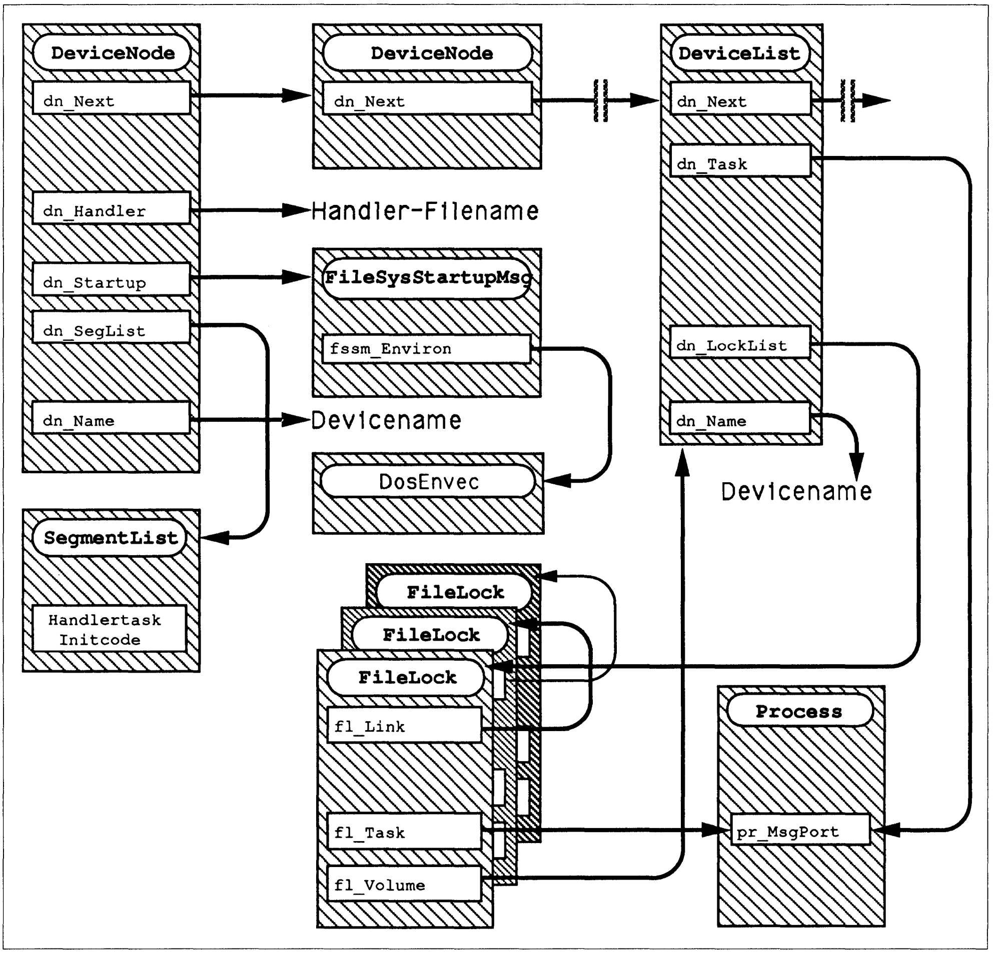


Abb. 6.2: Die DOS-Device-Liste

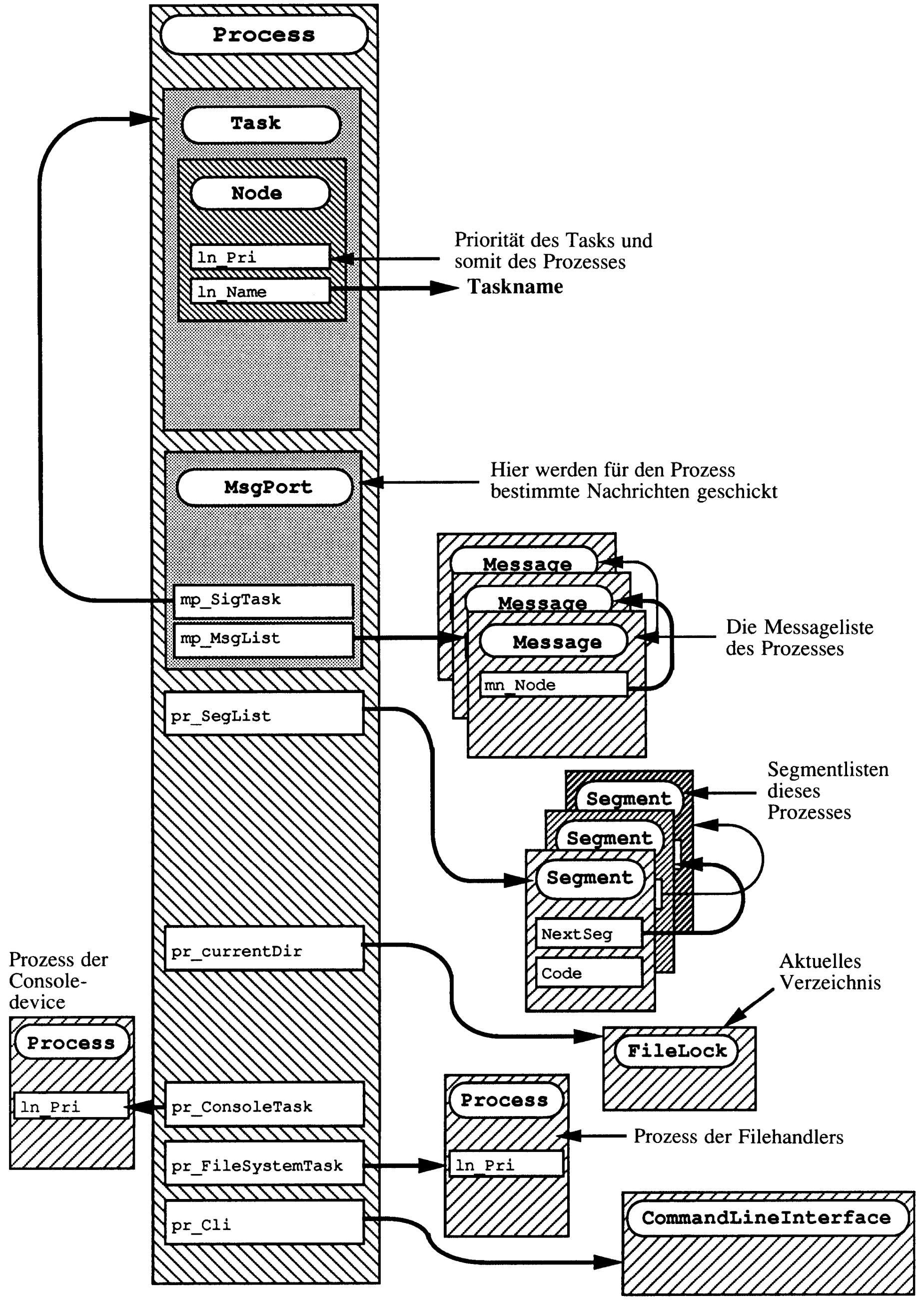


Abb. 6.3: Der Aufbau eines Prozesses

7 SONSTIGE LIBRARIES

7.1 Die Diskfont-Library

Die Aufgabe dieser Library ist es, das Laden, Speichern und Verwalten verschiedener Zeichensätze zu übernehmen. Beim Öffnen dieser Bibliothek wird die Adresse, die die Routine OpenLibrary zurückgibt, der Variablen DiskfontBase zugewiesen.

AvailFonts

Offset		C (libraries/diskfont.h)	Modula-2 (DiskFont)
Hex	**Dez**	struct AvailFonts	AvailFont =
		{	RECORD
0000	0	UWORD af_Type;	type: AvailFontTypeSet;
0002	2	struct TextAttr af_Attr;	attr: TextAttr;
000A	10	};	END;

Assembler Syntax: Strukturname = *AF*

Datenstrukturenreferenz »DiskFont«: *AvailFontsHeader*.

Routinenreferenz »DiskFont«: *AvailFonts*.

Beschreibung

Nach dem Aufruf der Routine *AvailFonts* wird in einer Reihe von *AvailFonts*-Datenstrukturen, die von einer *AvailFontsHeader*-Datenstruktur eingeleitet werden, die Informationen über die verfügbaren Zeichensätze gespeichert.

❑ **af_Type 0x0000 (0)** Diese 2-Byte-Marke gibt an, ob sich dieser Font im Speicher oder auf Diskette befindet.

C	Modula-2	Hex	Beschreibung
AFF_MEMORY	memory	0001	Der Zeichensatz befindet sich im Speicher.
AFF_DISK	disk	0002	Der Zeichensatz befindet sich auf Diskette.

❑ **af_Attr 0x0002 (2)**

Dies ist eine normale *TextAttr*-Datenstruktur aus »Graphics«. Sie beschreibt alle Eigenschaften der Zeichensätze.

Offset		**C (libraries/diskfont.h)**	**Modula-2 (DiskFont)**
Hex	**Dez**	struct AvailFontsHeader	AvailFontHeader =
		{	RECORD
0000	0	UWORD afh_NumEntries;	numEntries: CARDINAL;
0002	2	/* struct AvailFonts	(* af: ARRAY[0..numEntries-1] OF
		afh_AF[]; */	AvailFont; *)
		};	END;

AvailFontsHeader

Assembler Syntax: Strukturname = *AFH*

Datenstrukturenreferenz Keine.

Routinenreferenz »DiskFont«: *AvailFonts*.

Beschreibung

Diese Datenstruktur leitet nach Aufruf der *AvailFonts*-Routine die Liste der oben beschriebenen *AvailFonts*-Datenstrukturen ein.

❑ **afh_NumEntries 0x0000 (0)**

Gibt an, wie viele *AvailFonts*-Datenstrukturen in *afh_AF* zusammengefaßt sind.

❑ **afh_AF 0x0002 (2)**

Hier befinden sich so viele *AvailFonts*-Datenstrukturen, wie in *afh_NumEntries* angegeben.

Offset		C (libraries/diskfont.h)	Modula-2 (DiskFont)
Hex	Dez	struct DiskFontHeader	DiskFontHeader =
		{	RECORD
0000	0	struct Node dfh_DF;	df: Node;
000E	14	UWORD dfh_FileID;	fileId: CARDINAL;
0010	16	UWORD dfh_Revision;	revision: CARDINAL;
0012	18	LONG dfh_Segment;	segment: BPTR;
0016	22	char dfh_Name[32];	name: ARRAY [0..31] OF CHAR;
0036	54	struct TextFont dfh_TF;	tf: TextFont
006A	106	};	END;

DiskFontHeader

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Wird ein Diskfont mit Hilfe der Routine *LoadSeg* geladen, dann steht eine *DiskFontHeader*-Datenstruktur an der Adresse, die *LoadSeg* zurückgibt.

- **dfh_DF 0x0000 (0)** — Einbindung einer vollständigen *Node*-Datenstruktur aus der Exec-Library. Über solche Nodes sind alle Diskfonts verkettet.
- **dfh_FileID 0x000E (14)** — Hier steht immer der Wert: 0x0f80 (3968).
- **dfh_Revision 0x0010 (16)** — Gibt die Versionsnummer des Zeichensatzes an.
- **dfh_Segment 0x0012 (18)** — Ein Adreßzeiger auf das Segment nach dem Laden.
- **dfh_Name 0x0016 (22)** — Eine Zeichenkette bis 32 Zeichen Länge, die den Namen des Zeichensatzes enthält.
- **dfh_TF 0x0036 (54)** — Einbindung einer vollständigen *TextFont*-Datenstruktur aus der der Graphics-Library, die die genauen Daten des Zeichensatzes enthält.

Offset Hex	Offset Dez	C (libraries/diskfont.h)	Modula-2 (DiskFont)
		struct FontContents	FontContents =
		{	RECORD
0000	0	char fc_FileName[256];	fileName: ARRAY [0..255] OF CHAR;
0100	256	UWORD fc_YSize;	ySize: CARDINAL;
0102	258	UBYTE fc_Style;	style: FontStyleSet;
0103	259	UBYTE fc_Flags;	flags: FontFlagSet;
0104	260	};	END;

FontContens

Assembler Syntax: Strukturname = *FH*

Datenstrukturenreferenz »DiskFont« : *FontContensHeader*.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur beschreibt einige Eigenschaften eines sich auf Diskette befindenden Zeichensatzes und gibt an, unter welchen Namen dieser auf Diskette zu finden ist. Es werden mehrere solcher Datenstrukturen an eine *FontContentsHeader*-Datenstruktur angehängt.

❑ **fc_FileName 0x0000 (0)** Der Name (wenn nötig, mit Laufwerksnummer und Directory), unter dem die eigentlichen Daten dieses Zeichensatzes auf Diskette zu finden sind. Wird nur der Filename angegeben, so wird dieser in der *FONTS:* Directory gesucht.

❑ **fc_YSize 0x0100 (256)** Dieses Feld entspricht dem *ta_YSize*-Parameter der *TextAttr*-Datenstruktur.

❑ **fc_Style 0x0102 (258)** Dieses Feld entspricht dem *ta_Style*-Parameter der *TextAttr*-Datenstruktur.

❑ **fc_Flags 0x0103 (259)** Dieses Feld entspricht dem *ta_Flags*-Parameter der *TextAttr*-Datenstruktur.

Offset		**C (libraries/diskfont.h)**	**Modula-2 (DiskFont)**
Hex	**Dez**	struct FontContentsHeader	FontContentsHeader =
		{	RECORD
0000	0		UWORD fch_FileID; fileId: CARDINAL;
0002	2		UWORD fch_NumEntries; numEntries: CARDINAL;
0004	4		/* struct FontContens (*fc: ARRAY[0..numEntries-1] OF
		fch_FC[]; */	FontContents*)
		};	END;

FontContentsHeader

Assembler Syntax: Strukturname = *FCH*

Datenstrukturenreferenz Keine.

Routinenreferenz »DiskFont« : *DisposeFontContens*, *NewFontContents*.

Beschreibung

Diese Datenstruktur leitet eine Liste von *FontContents*-Datenstrukturen ein, die eine Familie ähnlicher Zeichensätze (die sich z.B. nur in der Größe unterscheiden) beschreiben. Diese Datenstruktur ist in dem File *<FONTNAME>.FONT* jedes Zeichensatzes enthalten (z.B. in *SAPHIRE.FONT*).

❑ **fch_FileID 0x0000 (0)** Dieser Wert identifiziert die Art des Files und muß immer 0x0f00 (3840) sein.

❑ **fch_NumEntries 0x0002 (2)** Anzahl der in *FC* folgenden *FontContents*-Datenstrukturen.

❑ **fch_FC 0x0004 (4)** Hier stehen so viele *FontContents*-Datenstrukturen, wie in *fch_NumEntries* angegeben.

7.2 Die Expansion-Library

Die Expansion-Library dient dazu, Erweiterungen in Form von Steckkarten in das Amiga-System einzubinden. Diese Library ist sehr Hardware-orientiert und daher besonders systemintern. Sie wirft auch für uns noch einige ungeklärte Fragen auf und ist in diesem Buch nur der Vollständigkeit halber mit eingeschlossen.

ConfigDev

Offset Hex	C Dez	(libraries/configvars.h)	Modula-2 (Expansion)
		struct ConfigDev	ConfigDev =
		{	RECORD
0000	0	struct Node cd_Node;	node:Node;
000E	14	UBYTE cd_Flags;	flags:UByte;
000F	15	UBYTE cd_Pad;	pad:UByte;
0010	16	struct ExpansionRom cd_Rom;	rom:ExpansionRom;
0020	32	APTR cd_BoardAddr;	boardAddr:ADDRESS;
0024	36	APTR cd_BoardSize;	boardSize:ADDRESS;
0028	40	UWORD cd_SlotAddr;	slotAddr:CARDINAL;
002A	42	UWORD cd_SlotSize;	slotSize:CARDINAL;
002C	44	APTR cd_Driver;	driver:ADDRESS;
0030	48	struct ConfigDev * cd_NextCD;	nextCD:ConfigDevPtr;
0034	52	ULONG cd_Unused[4];	unused:ARRAY [0..3] OF LONGINT;
0044	68	};	END;

Datenstrukturenreferenz »Expansion«: *ExpansionBase*, *CurrentBinding*.

Routinenreferenz »Expansion«: *AllocConfigDev*, *ConfigBoard*, *FindConfigDev*, *FreeConfigDev*, *ReadExpansionRom*, *RemConfigDev*.

Beschreibung

Diese Datenstruktur beschreibt eine externe Erweiterung und dient dazu, sie in das Amiga-System einzubinden. Zu diesem Zweck wird sie an die *ConfigBoard*-Routine übergeben, die sie in die entsprechende Liste der *ExpansionBase*-Datenstruktur einträgt.

- **cd_Node 0x0000 (0)** Über diese eingebundene *Node*-Datenstruktur werden mehrere *ConfigDev*-Datenstrukturen miteinander zu einer Liste verkettet.

- **cd_Flags 0x000E (14)** Diese Flags geben nähere Auskunft über den Zustand der Erweiterung. Folgende zwei Werte sind zugelassen:

C	Modula-2	Dez	Beschreibung
CDF_SHUTUP	shutup	1	Die Erweiterung ist ausgeschaltet.
CDF_CONFIGME	configMe	2	Die Erweiterung benötigt einen eigenenTreiber.

- **cd_Pad 0x000F (15)** Ein Füllbyte.
- **cd_Rom 0x0010 (16)** Eine eingebundene *ExpansionRom*-Datenstruktur zum Zugriff auf die Hardware der Erweiterung.
- **cd_BoardAddr 0x0020 (32)** Die Speicheradresse, an der sich die Erweiterung befindet.
- **cd_BoardSize 0x0024 (36)** Die Größe der Erweiterung (natürlich nicht in Quadratzentimetern, sondern in Bytes).
- **cd_SlotAddr 0x0028 (40)** Die Nummer des Slots, in dem sich die Erweiterung befindet.
- **cd_SlotSize 0x002A (42)** Die Anzahl der Slots, die durch diese Erweiterung belegt werden.
- **cd_Driver 0x002C (44)** Ein Adreßzeiger auf die *Node*-Datenstruktur des zugehörigen Treibers.
- **cd_NextCD 0x0030 (48)** Ein Adreßzeiger auf die zu konfigurierenden Treiber dieser Erweiterung.
- **cd_Unused 0x0034 (52)** Speicherplatz für zukünftige Erweiterungen dieser Datenstruktur.

Offset	C	(libraries/configvars.h)	Modula-2 (Expansion)
Hex	Dez	struct CurrentBinding	CurrentBinding =
		{	RECORD
0000	0	struct ConfigDev * cb_ConfigDev;	configDev:ConfigDevPtr;
0004	4	UBYTE * cb_FileName;	fileName:ADDRESS;
0008	8	UBYTE * cb_ProductString;	productString:ADDRESS;
000C	12	UBYTE ** cb_ToolTypes;	toolTypes:ADDRESS;
0010	16	};	END;

CurrentBinding

Datenstrukturenreferenz Keine.

Routinenreferenz »Expansion«: *GetCurrentBinding*, *SetCurrentBinding*.

Beschreibung

Diese Datenstruktur beschreibt die aktuelle Konfiguration einer Erweiterung.

❑ **cb_ConfigDev 0x0000 (0)** Ein Adreßzeiger auf die Liste der zugehörigen *ConfigDev*-Datenstrukturen.

❑ **cb_FileName 0x0004 (4)** Ein Adreßzeiger auf den Namen des Treibers.

❑ **cb_ProductString 0x0008 (8)** Ein Adreßzeiger auf den Namen (oder sonstige ID) der Erweiterung.

❑ **cb_ToolTypes 0x000C (12)** Ein Adreßzeiger auf die in einer *DiskObjekt*-Datenstruktur enthaltenen Startup-Informationen des Treibers (siehe »Workbench«).

DiagArea

Offset	C	(libraries/configregs.h)	Modula-2 (Expansion)
Hex	**Dez**	struct DiagArea	DiagArea =
		{	RECORD
0000	0	UBYTE da_Config;	config:UByte;
0001	1	UBYTE da_Flags;	flags:UByte;
0002	2	UWORD da_Size;	size:CARDINAL;
0004	4	UWORD da_DiagPoint;	diagPoint:CARDINAL;
0006	6	UWORD da_BootPoint;	bootPoint:CARDINAL;
0008	8	UWORD da_Name;	name:CARDINAL;
000A	10	UWORD da_Reserved01;	reserved01:CARDINAL;
000C	12	UWORD da_Reserved02;	reserved02:CARDINAL
000E	14	};	END;

Datenstrukturenreferenz »Expansion«: *ExpansionRom.*

Routinenreferenz Keine.

Beschreibung

Falls das *ERTB_DIAGVALID*-Bit des *er_Type*-Parameters der *ExpansionRom*-Datenstruktur gesetzt ist, zeigt das *er_InitDiagVec*-Feld auf eine solche, im ROM der Erweiterung enthaltene Datenstruktur. Dabei handelt es sich um einen Offset von der Basis der Erweiterung. Sie gibt weiterhin an, was bei einem Bootvorgang oder einem Fehler mit der Erweiterung getan werden soll.

❑ **da_Config 0x0000 (0)** Dieses Byte gibt an, mit welcher Wortbreite diese Datenstruktur im ROM der Erweiterung vorhanden ist:

C	Modula-2	Hex	Beschreibung
DAC_NIBBLEWIDE	nibbleWide	00	Die Wortbreite beträgt 4 Bit.
DAC_BYTEWIDE	byteWide	40	Die Wortbreite beträgt 8 Bit.
DAC_WORDWIDE	wordWide	80	Die Wortbreite beträgt 16 Bit.
DAC_BUSWIDTH	busWidth	C0	Die Wortbreite beträgt 32 Bit.

❑ **da_Flags 0x0001 (1)**

Folgende Flags geben die Funktion der Datenstruktur an (wann sie benutzt wird):

C	Modula-2	Hex	Beschreibung
DAC_NEVER	never	00	Sie wird nie benutzt.
DAC_CONFIGTIME	configTime	10	Sie wird beim Konfigurieren benutzt.
DAC_BINDTIME	bindTime	20	Sie wird beim »Binding« benutzt.
DAC_BOOTTIME	bootTime	20	Sie wird beim Booten benutzt.

❑ **da_Size 0x0002 (2)**

Die Größe, die diese Datenstruktur im ROM der Erweiterung einnimmt.

❑ **da_DiagPoint 0x0004 (4)**

Ein Adreßzeiger auf eine eventuell vorhandene Fehlerbehandlungsroutine der Erweiterung oder null.

❑ **da_BootPoint 0x0006 (6)**

Ein Adreßzeiger auf eine eventuell vorhandene Boot-Routine der Erweiterung oder null.

❑ **da_Name 0x0008 (8)**

Ein Adreßzeiger auf einen evtl. im ROM der Erweiterung vorhandenen Namen, aber was für einen?

❑ **da_Reserved01 0x000A (10)**

Reserviertes Wort für internen Gebrauch, auf null setzen.

❑ **da_Reserved02 0x000C (12)**

Reserviertes Wort für internen Gebrauch, auf null setzen.

ExpansionBase

Offset	C	(libraries/expansionbase.h)
Hex	Dez	struct ExpansionBase
		{
0000	0	struct Library LibNode;
0022	34	UBYTE Flags;
0023	35	UBYTE pad;
0024	36	APTR ExecBase;
0028	40	APTR SegList;
002C	44	struct CurrentBinding CurrentBinding;
003C	60	struct List BoardList;
004A	74	struct List MountList;
0058	88	UBYTE AllocRTable[256];
0158	344	struct SignalSemaphore BindSemaphore;
0186	390	struct Interrupt Int2List;
019C	412	struct Interrupt Int6List;
01B2	434	struct Interrupt Int7List;
01C8	456	};

Assembler	Präfix:	*eb_*
	Syntax:	Der Parameter *LibNode* ist nicht definiert.
Datenstrukturenreferenz	Keine.	
Routinenreferenz	Keine.	

Beschreibung

Dies ist eine erweiterte *Library*-Datenstruktur, die die Expansion-Library beschreibt. Diese systeminterne Datenstruktur ist nur in neueren Versionen der verschiedenen Compiler implementiert.

- **LibNode 0x0000 (0)** — Eine eingebundene *Library*-Datenstruktur zur Einbindung der Expansion-Library in das System.
- **Flags 0x0022 (34)** — Systeminterne Flags.
- **pad 0x0023 (35)** — Ein Füllbyte ohne weitere Bedeutung.
- **ExecBase 0x0024 (36)** — Ein Adreßzeiger auf die *ExecBase*-Datenstruktur.
- **SegList 0x0028 (40)** — Eine Segmentliste, wer weiß wovon?
- **CurrentBinding 0x002C (44)** — Ein Adreßzeiger auf eine *CurrentBinding*-Datenstruktur, die wohl die momentan aktiven Erweiterungen spezifiziert.
- **BoardList 0x003C (60)** — Der Kopf der Liste aller Erweiterungen.
- **MountList 0x004A (74)** — Eine Vermutung: Der Kopf der Liste all der Erweiterungen, die »gemountet« werden müssen.
- **AllocRTable 0x0058 (88)** — Offensichtlich eine Tabelle... Anscheinend eine für alle Slots (256 benötigten allerdings einen neuartigen 2,50 m breiten Amiga?).
- **BindSemaphore 0x0158 (344)** — Eine eingebundene *Semaphore*-Datenstruktur, die anscheinend den Zugriff auf den *CurrentBinding*-Parameter koordinieren soll.
- **Int2List 0x0186 (390)** — ???
- **Int6List 0x019C (412)** — ???
- **Int7List 0x01B2 (434)** — ???

7

ExpansionControl

Offset Hex	C Dez	(libraries/configregs.h)	Modula-2 (Expansion)
		struct ExpansionControl	ExpansionControl =
		{	RECORD
0000	0	UBYTE ec_Interrupt;	interrupt:UByte;
0001	1	UBYTE ec_Reserved11;	reserved11:UByte;
0002	2	UBYTE ec_BaseAddress;	baseAddress:UByte;
0003	3	UBYTE ec_Shutup;	shutup:UByte;
0004	4	UBYTE ec_Reserved14;	reserved14:UByte;
0005	5	UBYTE ec_Reserved15;	reserved15:UByte;
0006	6	UBYTE ec_Reserved16;	reserved16:UByte;
0007	7	UBYTE ec_Reserved17;	reserved17:UByte;
0008	8	UBYTE ec_Reserved18;	reserved18:UByte;
0009	9	UBYTE ec_Reserved19;	reserved19:UByte;
000A	10	UBYTE ec_Reserved1a;	reserved1a:UByte;
000B	11	UBYTE ec_Reserved1b;	reserved1b:UByte;
000C	12	UBYTE ec_Reserved1c;	reserved1c:UByte;
000D	13	UBYTE ec_Reserved1d;	reserved1d:UByte;
000E	14	UBYTE ec_Reserved1e;	reserved1e:UByte;
000F	15	UBYTE ec_Reserved1f;	reserved1f:UByte;
0010	16	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Laut [KRA] dient diese Datenstruktur zur logischen Beschreibung des Kontrollregister-Bereiches einer Erweiterung. Sie kann den physikalischen Aufbau nur näherungsweise wiedergeben. Die genaue Bedeutung der einzelnen Felder wird wohl von Erweiterung zu Erweiterung variieren?

- **ec_Interrupt 0x0000 (0)** Dieser Wert entspricht dem Interrupt-Register der Erweiterung.
- **ec_Reserved11 0x0001 (1)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_BaseAddress 0x0002 (2)** Hängt mit der Veränderung der Basisadresse innerhalb der Erweiterung des Systems zusammen ???
- **ec_Shutup 0x0003 (3)** Soll vielleicht zum Ein- und Ausschalten der Erweiterung benutzt werden?
- **ec_Reserved14 0x0004 (4)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved15 0x0005 (5)** Ein für den internen Gebrauch reserviertes Byte.

- **ec_Reserved16 0x0006 (6)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved17 0x0007 (7)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved18 0x0008 (8)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved19 0x0009 (9)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved1a 0x000A (10)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved1b 0x000B (11)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved1c 0x000C (12)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved1d 0x000D (13)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved1e 0x000E (14)** Ein für den internen Gebrauch reserviertes Byte.
- **ec_Reserved1f 0x000F (15)** Ein für den internen Gebrauch reserviertes Byte.

ExpansionRom

Offset	C	(libraries/configregs.h)	Modula-2 (Expansion)
Hex	**Dez**	struct ExpansionRom	ExpansionRom =
		{	RECORD
0000	0	UBYTE er_Type;	type:UByte;
0001	1	UBYTE er_Product;	product:UByte;
0002	2	UBYTE er_Flags;	flags:UByte;
0003	3	UBYTE er_Reserved03;	reserved03:UByte;
0004	4	UWORD er_Manufacturer;	manufacturer:CARDINAL;
0006	6	ULONG er_SerialNumber;	serialNumber:LONGCARD;
000A	10	UWORD er_InitDiagVec;	initDiagVec:CARDINAL;
000C	12	UBYTE er_Reserved0c;	reserved0c:UByte;
000D	13	UBYTE er_Reserved0d;	reserved0d:UByte;
000E	14	UBYTE er_Reserved0e;	reserved0e:UByte;
000F	15	UBYTE er_Reserved0f;	reserved0f:UByte
0010	16	};	END;

Datenstrukturenreferenz »Expansion«: *ConfigDev.*

Routinenreferenz Keine.

Beschreibung

Laut [KRA] dient diese Datenstruktur zur logischen Beschreibung des ROM-Bereiches einer Erweiterung. Sie kann den physikalischen Aufbau nur näherungsweise wiedergeben. Die genaue Bedeutung der einzelnen Felder wird wohl von Erweiterung zu Erweiterung variieren.

- **er_Type 0x0000 (0)** Folgende Bits sind für dieses Feld definiert:

C	Modula-2	Dez	Beschreibung
ERT_DIAGVALID	diagValid	4	Wenn dieses Bit gesetzt ist, muß in *er_InitDiagVec* ein Adreßzeiger auf eine korrekt initialisierte *DiagArea*-Datenstruktur stehen.

- **er_Product 0x0001 (1)** Nähere Angaben zu dem Produkt, also der Erweiterung selbst.

C	Modula-2	Bit	Beschreibung
ERFB_MEMSPACE	memSpace	7	Die Erweiterung muß innerhalb der unteren acht Mbyte des Adreßbereichs angesiedelt sein. Ist in diesem Bereich jedoch beliebig verschiebbar.
ERFB_NOSHUTUP	noShutup	6	Die Erweiterung ist nicht abschaltbar.

7

- **er_Flags 0x0002 (2)** Folgende Bits sind für dieses Feld definiert:
- **er_Reserved03 0x0003 (3)** Ein für den internen Gebrauch reserviertes Byte.
- **er_Manufacturer 0x0004 (4)** Nähere Angaben über den Hersteller der Erweiterung.
- **er_SerialNumber 0x0006 (6)** Die Seriennummer der Erweiterung.
- **er_InitDiagVec 0x000A (10)** Falls in *er_Types* das *ERT_DIAGVALID*-Bit gesetzt ist, ist dies ein Offset, der die Lage einer *DiagArea*-Datenstruktur innerhalb des Erweiterungs-ROMs angibt. So können Routinen der Erweiterung in das System eingebunden werden.
- **er_Reserved0c 0x000C (12)** Ein für den internen Gebrauch reserviertes Byte.
- **er_Reserved0d 0x000D (13)** Ein für den internen Gebrauch reserviertes Byte.
- **er_Reserved0e 0x000E (14)** Ein für den internen Gebrauch reserviertes Byte.
- **er_Reserved0f 0x000F (15)** Ein für den internen Gebrauch reserviertes Byte.

7.3 Die Hardware-Library

Die Datenstrukturen dieser Library dienen dazu, den Zugriff auf die Hardware zu erleichtern. Die Komponenten der Datenstrukturen entsprechen den Registern der verschiedenen Chips des Amiga. Daher ist dieses Kapitel eigentlich eine Hardware-Beschreibung.

bltnode

Offset Hex	Offset Dez	C (hardware/blit.h)	Modula-2 (Hardware)
		struct bltnode	Bltnode =
		{	RECORD
0000	0	struct bltnode *n;	n:BltnodePtr;
0004	4	int (*function)();	function:ADDRESS;
0008	8	char stat;	stat:CHAR;
000A	10	short blitsize;	blitsize:INTEGER;
000C	12	short beamsync;	beamsync:INTEGER;
000E	14	int (*cleanup)();	cleanup:ADDRESS;
0012	18	};	END;

Assembler Präfix: *bn_*

Datenstrukturenreferenz »Graphics« : *bltnode.* »Hardware« : *GfxBase.*

Routinenreferenz »Graphics« : *QBlit, QBSBlit.*

Beschreibung

Die *bltnode*-Datenstruktur wird zur Beschreibung von eigenständigen Blitter-Jobs verwendet, die in die zuständigen Job-Listen der GraphicsBase-Datenstruktur (siehe »Graphics«) eingetragen werden.

- **n 0x0000 (0)** Ein Adreßzeiger auf die *bltnode*-Datenstruktur des nächsten Blitter-Jobs in dieser Liste. Über die n-Zeiger werden sämtliche Blitter-Jobs einer Job-Liste vorwärts verkettet.

- **function 0x0004 (4)** Ein Adreßzeiger auf den Codebereich eines Unterprogramms, das ausgeführt wird, sobald dieser Blitter-Job an der Reihe ist. Das Unterprogramm muß mit einer RTS-Instruktion verlassen werden, wobei in Register D0 ein ganzzahliges Funktionsergebnis zurückgegeben wird. Ist der Ergebniswert ungleich null, so wird das Unterprogramm ein weiteres Mal aufgerufen (und zwar so lange, bis es eine Null als Ergebnis liefert!). In Assembler geschriebene Unterroutinen können folgende Registerkonventionen ausnutzen: Beim Aufruf befindet sich in Register A0 ein Adreßzeiger auf den Hardware-Registerbereich (Zugriff durch Offset via A0!) und in Register A1 ein Adreßzeiger auf die bltnode-Datenstruktur des aktuellen Blitter-Jobs.

- **stat 0x0008 (8)** Ein Zeichen, dessen Wert angibt, ob die *cleanup*-Routine dieses Blitter-Jobs am Ende ausgeführt werden soll oder nicht. Das ist der Fall, wenn der ASCII-Code des Zeichens 0x40 = 64 entspricht. Anderenfalls wird die cleanup-Routine nicht ausgeführt, so daß sie mit NULL initialisiert werden kann.
- **blitSize 0x000A (10)** Eine positive Ganzzahl, deren Wert die Ausmaße des durch diesen Blitter-Job zu bearbeitenden Datenbereichs im Format des *bltsize* Registers (siehe *Custom*) angibt.
- **beamSync 0x000C (12)** Eine positive Ganzzahl, deren Wert angibt, bei welcher Position des Elektronenstrahls die *function*-Routine eines elektronenstrahlsynchronisierten Blitter-Jobs aufgerufen werden soll.
- **cleanup 0x000E (14)** Die Adresse eines Unterprogramms, das wahlweise nach Erledigung des eigentlichen Blitter-Jobs aufgerufen wird, um spezielle Status- und Aufräumarbeiten zu erledigen.

CIA

7

Offset		**C (hardware/cia.h)**	**Modula-2 (Hardware)**	
Hex	**Dez**	struct CIA {	CIAA= RECORD	CIAB = RECORD
0000	0	UBYTE ciapra;	pra:CiaaPraFlagSet;	pra:CiabPraFlagSet;
0001	1	UBYTE pad0[0xff];	pad0:Pad;	pad0:Pad;
0100	256	UBYTE ciaprb;	prb:CiaaPrbFlagSet;	prb:CiabPrbFlagSet;
0101	257	UBYTE pad1[0xff];	pad1:Pad;	pad1:Pad;
0200	512	UBYTE ciaddra;	ddra:CiaaPraFlagSet;	ddra:CiabPraFlagSet;
0201	513	UBYTE pad2[0xff];	pad2:Pad;	pad2:Pad;
0300	768	UBYTE ciaddrb;	ddrb:CiaaPrbFlagSet;	ddrb:CiabPrbFlagSet;
0301	769	UBYTE pad3[0xff];	pad3:Pad;	pad3:Pad;
0400	1024	UBYTE ciatalo;	talo:UByte;	talo:UByte;
0401	1025	UBYTE pad4[0xff];	pad4:Pad;	pad4:Pad;
0500	1280	UBYTE ciatahi;	tahi:UByte;	tahi:UByte;
0501	1281	UBYTE pad5[0xff];	pad5:Pad;	pad5:Pad;
0600	1536	UBYTE ciatblo;	tblo:UByte;	tblo:UByte;
0601	1537	UBYTE pad6[0xff];	pad6:Pad;	pad6:Pad;
0700	1792	UBYTE ciatbhi;	tbhi:UByte;	tbhi:UByte;
0701	1793	UBYTE pad7[0xff];	pad7:Pad;	pad7:Pad;
0800	2048	UBYTE ciatodlow;	todlow:UByte;	todlow:UByte;
0801	2049	UBYTE pad8[0xff];	pad8:Pad;	pad8:Pad;
0900	2304	UBYTE ciatodmid;	todmid:UByte;	todmid:UByte;
0901	2305	UBYTE pad9[0xff];	pad9:Pad;	pad9:Pad;
0A00	2560	UBYTE ciatodhi;	todhi:UByte;	todhi:UByte;

Offset		C (hardware/cia.h)	Modula-2 (Hardware)	
Hex	Dez			
0A01	2561	UBYTE pad10[0xff];	pad10:Pad;	pad10:Pad;
0B00	2816	UBYTE unusedreg;	unusedreg:ShortSet;	unusedreg:ShortSet;
0B01	2817	UBYTE pad11[0xff];	pad11:Pad;	pad11:Pad;
0C00	3072	UBYTE ciasdr;	sdr:ShortSet;	sdr:ShortSet;
0C01	3073	UBYTE pad12[0xff];	pad12:Pad;	pad12:Pad;
0D00	3328	UBYTE ciaicr;	icr:CiaIcrFlagSet;	icr:CiaIcrFlagSet;
0D01	3329	UBYTE pad13[0xff];	pad13:Pad;	pad13:Pad;
0E00	3584	UBYTE ciacra;	cra:CiaCraFlagSet;	cra:CiaCraFlagSet;
0E01	3585	UBYTE pad14[0xff];	pad14:Pad;	pad14:Pad;
0F00	3840	UBYTE ciacrb;	crb:CiaCrbFlagSet;	crb:CiaCrbFlagSet;
0F01	3841	};	END;	END;

Assembler Es sind nur die *ciaxxx*-Offsets definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Die CIA-Datenstruktur stellt nichts anderes als eine Beschreibung der Hardware-Register des im Amiga für die Kommunikation mit der Peripherie verwendeten CIA-Bausteins 8520 dar. Die Abkürzung »CIA« steht dabei für »complex interface adapter«, was man sinngemäß mit »vielseitiger Schnittstellenbaustein« übersetzen kann. Der CIA 8520 ist dem im C64 verwendeten CIA 6526 zum Verwechseln ähnlich, lediglich die *TODxx*-Register wurden beim 8520 geringfügig verändert: Sie fungieren hier als reine 24-Bit-Zähler, wohingegen im 6526 an dieser Stelle eine Echtzeituhr »time of day« vorhanden war. Den CIA-Baustein 8520 und seine Register in ihrer allgemeinen Funktion präsentiert Ihnen die nachfolgende Auflistung:

Register	Name	Bedeutung
00 0x00	PRA	Datenregister für Port A
01 0x01	PRB	Datenregister für Port B
02 0x02	DDRA	Datenrichtungsregister für Port A
03 0x03	DDRB	Datenrichtungsregister für Port B
04 0x04	TALO	Niederwertiges Byte (Bits 0 bis 7) für Timer A
05 0x05	TAHI	Höherwertiges Byte (Bits 8 bis 15) für Timer A
06 0x06	TBLO	Niederwertiges Byte (Bits 0 bis 7) für Timer B
07 0x07	TBHI	Höherwertiges Byte (Bits 8 bis 15) für Timer B
08 0x08	TODLO	Niederwertiges Byte (Bits 0 bis 7) des Event-Zählers
09 0x09	TODMI	Mittleres Byte (Bits 8 bis 15) des Event-Zählers
10 0x0A	TODHI	Höherwertiges Byte (Bits 16 bis 23) des Event-Zählers
11 0x0B	-/-	

Register	Name	Bedeutung
12 0x0C	SDR	Datenregister des seriellen Ports
13 0x0D	ICR	Interrupt-Kontrollregister
14 0x0E	CRA	Kontrollregister für Timer A
15 0x0F	CRB	Kontrollregister für Timer B

Im Amiga verrichten zwei 8520-CIA-Bausteine ihre Arbeit: der CIA-A mit Basisadresse 0x00BFE001 = 12574721 und der CIA-B mit Basisadresse 0x00BFD000 = 12570624. Aus speichertechnischen Gründen liegen die Register eines CIA-Bausteins im Amiga jeweils 256 Byte voneinander entfernt. Deshalb erhält man die tatsächliche Adresse eines CIA-Registers durch Addition von 0x0100*R = 256*R zur Basisadresse des CIA-Bausteins (dabei steht R für die Nummer des gewünschten Registers). In Modula-2 sind solche Umrechnungen allerdings überflüssig, weil hier im Modul Hardware bereits zwei Variablen *ciaa* und *ciab* der Typen CIAA bzw. CIAB ab den absoluten Adressen 0x00BFE001 sowie 0x00BFD000 deklariert sind. In »C« umgeht man die Rechnerei durch eine kleine Zeigermanipulation:

```
struct CIA *ciaa, *ciab;
.....
ciaa = (CIA)* 0x00BFE001;      /*  Zeiger auf Registersatz richten    */
ciab = (CIA)* 0x00BFD000;      /*  dto.  */
```

Aufgrund der unterschiedlichen Belegung und Benutzung der einzelnen Register der beiden CIA-Bausteine werden für diese unter Modula-2 im Modul Hardware zwei unterschiedliche Typen namens CIAA und CIAB vereinbart. In »C« existieren solche Strukturen leider nicht. Wie die zwei CIA-Bausteine im einzelnen in das Gesamtsystem des Amiga eingebunden sind, erfahren Sie im folgenden.

CIA-A

Adresse	Name	D7	D6	D5	D4	D3	D2	D1	D0	(Bedeutung)
0x00BFE001	PRA	/FIR1	/FIR0	/RDY	/TK0	/WPRO	/CHNG	/LED	/OVL	
0x00BFE101	PRB	8-Bit I/O-Register für den Parallelport (Centronics)								
0x00BFE201	DDRA	0	0	0	0	0	0	1	1	
0x00BFE301	DDRB	Hiermit können die einzelnen Leitungen des Parallelports als Ein- oder Ausgänge betrieben werden.								
0x00BFE401	TALO	Timer A des CIA-A wird nur für den Datentransfer per								
0x00BFE501	TAHI	serieller Schnittstelle benötigt. Ansonsten ist er frei.								
0x00BFE601	TBLO	Timer B des CIA-A dient der Synchronisation des Blitters								
0x00BFE701	TBHI	mit dem Elektronenstrahl. Daher wird Timer B nur für QBS Blitter-Jobs (siehe *GfxBase*-Datenstruktur aus »Graphics«) benötigt. Ansonsten ist er frei verfügbar.								
0x00BFE801	TODLO	Der Event-Zähler des CIA-A registriert die Frequenz-Impulse								
0x00BFE901	TODMI	(»Ticks«) des Netzteils. Die in Deutschland übliche 50 Hz								
0x00BFEA01	TODHI	Versorgungsspannung ergibt ein Inkrement von 50 je Sekunde.								
0x00BFEB01	-/-									

Adresse	Name	Bedeutung
0x00BFEC01	SDR	Datenregister für den Empfang der Tastaturcodes
0x00BFED01	ICR	Interrupt-Kontrollregister des CIA-A
0x00BFEE01	CRA	Kontrollregister für Timer A des CIA-A
0x00BFEF01	CRB	Kontrollregister für Timer B des CIA-A

Wichtig für die Systemsteuerung ist, daß der CIA-A einen INT2*Interrupt erzeugen kann. Die einzelnen Bits im Portregister A des CIA-A haben folgende Bedeutung:

Bit	Name	Bedeutung
07	FIR1	Eingabebit: Wird der Feuerknopf des Mouse-Ports 1 gerade gedrückt?
06	FIR0	Eingabebit: Wird der Feuerknopf des Mouse-Ports 0 gerade gedrückt?
05	RDY	Eingabebit: Ist das ausgewählte Laufwerk bereit?
04	TK0	Eingabebit: Ist der Kopf des Laufwerks gerade über Spur 0?
03	WPRO	Eingabebit: Ist die Diskette im Laufwerk schreibgeschützt?
02	CHNG	Eingabebit: Hat ein Diskettenwechsel im Laufwerk stattgefunden?
01	LED	Ausgabebit: Steuerung der Power-LED (0=hell, 1=dunkel)
00	OVL	Ausgabebit: Speicherüberlaufsbit »memory overlay« (nicht ändern!)

CIA-B

Adresse	Name	D7	D6	D5	D4	D3	D2	D1	D0	(Bedeutung)
0x00BFD000	PRA	/DIR	/RTS	/CD	/CTS	/DSR	/SEL	/POUT	/BUSY	
0x00BFD100	PRB	/MTR	/SEL3	/SEL2	/SEL1	/SEL0	/SIDE	/DIR	/STEP	
0x00BFD200	DDRA	1	1	0	0	0	0	0	0	
0x00BFD300	DDRB	1	1	1	1	1	1	1	1	
0x00BFD400	TALO									Timer A des CIA-B wird vom Betriebssystem zur Kommunikation
0x00BFD500	TAHI									mit der Tastatur benutzt.
0x00BFD600	TBLO									Timer B des CIA-B dient dem Betriebssystem als virtuelles
0x00BFD700	TBHI									»timer device«, das für Interrupts und die Task-Umschaltung benötigt wird.
0x00BFD800	TODLO									Der Event-Zähler des CIA-B registriert die horizontalen
0x00BFD900	TODMI									Synchronisationsimpulse. Im PAL-Betrieb sind das normaler-
0x00BFDA00	TODHI									weise genau 15625 pro Sekunde (PAL-Zeilenfrequenz!).
0x00BFDB00	-/-									
0x00BFDC00	SDR									(bisher nicht benutzt)
0x00BFDD00	ICR									Interrupt-Kontrollregister des CIA-B
0x00BFDE00	CRA									Kontrollregister für Timer A des CIA-B
0x00BFDF00	CRB									Kontrollregister für Timer B des CIA-B

Wichtig für die Systemsteuerung ist, daß der CIA-B einen INT6*Interrupt erzeugen kann. Die einzelnen Bits im Portregister A des CIA-B haben folgende Bedeutung:

Bit	Name	Bedeutung
07	DTR	Ausgabebit: DTR-Steuersignal der seriellen Schnittstelle
06	RTS	Ausgabebit: RTS-Steuersignal der seriellen Schnittstelle
05	CD	Eingabebit: CD-Signal der seriellen Schnittstelle
04	CTS	Eingabebit: CTS-Signal der seriellen Schnittstelle
03	DSR	Eingabebit: DSR-Signal der seriellen Schnittstelle
02	SEL	Eingabebit: SELECT-Signal der parallelen Schnittstelle
01	POUT	Eingabebit: PAPER OUT-Signal der parallelen Schnittstelle
00	BUSY	Eingabebit: BUSY-Signal der parallelen Schnittstelle

Die Bedeutung der einzelnen Bits im Portregister B des CIA-B entnehmen Sie bitte der nachfolgenden Aufstellung:

Bit	Name	Bedeutung
07	MTR	Ausgabebit: MOTOR-Steuersignal für das Diskettenlaufwerk
06	SEL3	Ausgabebit: Auswahlsignal »drive select« für Laufwerk 3
05	SEL2	Ausgabebit: Auswahlsignal »drive select« für Laufwerk 2
04	SEL1	Ausgabebit: Auswahlsignal »drive select« für Laufwerk 1
03	SEL0	Ausgabebit: Auswahlsignal »drive select« für Laufwerk 0
02	SIDE	Ausgabebit: Auswahlsignal »side select« ans Diskettenlaufwerk
01	DIR	Ausgabebit: DIRECTION-Steuersignal für das Diskettenlaufwerk
00	STEP	Ausgabebit: STEP-Signal für das Diskettenlaufwerk

- **ciapra 0x0000 (0)**
  Adresse: 0x00BFE001 = 12574721 für CIA-A,
  0x00BFD000 = 12570624 für CIA-B
  Funktion: *PRA*-Register

- **pad0 0x0001 (1)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciaprb 0x0100 (256)**
  Adresse: 0x00BFE101 = 12574977 für CIA-A,
  0x00BFD100 = 12570880 für CIA-B
  Funktion: *PRB*-Register

- **pad1 0x0101 (257)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciaddra 0x0200 (512)**
  Adresse: 0x00BFE201 = 12575233 für CIA-A,
  0x00BFD200 = 12571136 für CIA-B
  Funktion: *DDRA*-Register

- **pad2 0x0201 (513)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciaddrb 0x0300 (768)**
  Adresse: 0x00BFE301 = 12575489 für CIA-A,
  0x00BFD300 = 12571392 für CIA-B
  Funktion: *DDRB*-Register

- **pad3 0x0301 (769)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatalo 0x0400 (1024)**
  Adresse: 0x00BFE401 = 12575745 für CIA-A,
  0x00BFD400 = 12571648 für CIA-B
  Funktion: *TALO*-Register

- **pad4 0x0401 (1025)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatahi 0x0500 (1280)**
  Adresse: 0x00BFE501 = 12576001 für CIA-A,
  0x00BFD500 = 12571904 für CIA-B
  Funktion: *TAHI*-Register

- **pad5 0x0501 (1281)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatblo 0x0600 (1536)**
  Adresse: 0x00BFE601 = 12576257 für CIA-A,
  0x00BFD600 = 12572160 für CIA-B
  Funktion: *TBLO*-Register

- **pad6 0x0601 (1537)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatbhi 0x0700 (1792)**
  Adresse: 0x00BFE701 = 12576513 für CIA-A,
  0x00BFD700 = 12572416 für CIA-B
  Funktion: *TBHI*-Register

- **pad7 0x0701 (1793)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatodlow 0x0800 (2048)**
  Adresse: 0x00BFE801 = 12576769 für CIA-A,
  0x00BFD800 = 12572672 für CIA-B
  Funktion: *TODLO*-Register

- **pad8 0x0801 (2049)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatodmid 0x0900 (2304)**
  Adresse: 0x00BFE901 = 12577025 für CIA-A,
  0x00BFD900 = 12572928 für CIA-B
  Funktion: *TODMI*-Register

- **pad9 0x0901 (2305)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciatodhi 0x0A00 (2560)**
  Adresse: 0x00BFEA01 = 12577281 für CIA-A,
  0x00BFDA00 = 12573184 für CIA-B
  Funktion: *TODHI*-Register

- **pad10 0x0A01 (2561)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **unusedreg 0x0B00 (2816)**
  (unbenutztes CIA-Register)

- **pad11 0x0B01 (2817)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciasdr 0x0C00 (3072)**
  Adresse: 0x00BFEC01 = 12577793 für CIA-A,
  0x00BFDC00 = 12573696 für CIA-B
  Funktion: *SDR*-Register

- **pad12 0x0C01 (3073)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciaicr 0x0D00 (3328)**
  Adresse: 0x00BFED01 = 12578049 für CIA-A,
  0x00BFDD00 = 12573952 für CIA-B
  Funktion: *ICR*-Register

- **pad13 0x0D01 (3329)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciacra 0x0E00 (3584)**
  Adresse: 0x00BFEE01 = 12578305 für CIA-A,
  0x00BFDE00 = 12574208 für CIA-B
  Funktion: *CRA*-Register

- **pad14 0x0E01 (3585)**
  Dummy-Komponente zur Adreßfüllung zwischen den CIA-Registern.

- **ciacrb 0x0F00 (3840)**
  Adresse: 0x00BFEF01 = 12578561 für CIA-A,
  0x00BFDF00 = 12574464 für CIA-B
  Funktion: *CRB*-Register

Custom

Offset		C (hardware/custom.h)	Modula-2 (Hardware)
Hex	Dez	struct Custom	Custom =
		{	RECORD
0000	0	UWORD bltddat;	bltddat:CARDINAL;
0002	2	UWORD dmaconr;	dmaconr:DmaFlagSet;
0004	4	UWORD vposr;	vposr:LONGCARD;
0006	6	UWORD vhposr;	
0008	8	UWORD dskdatr;	dskdatr:CARDINAL;
000A	10	UWORD joy0dat;	joy0dat:Coord;
000C	12	UWORD joy1dat;	joy1dat:Coord;
000E	14	UWORD clxdat;	clxdat:Collison Flag Set;
0010	16	UWORD adkconr;	adkconr:Adk Flag Set;
0012	18	UWORD pot0dat;	pot0dat:Coord;
0014	20	UWORD pot1dat;	pot1dat:Coord;
0016	22	UWORD potinp;	potinp:Pot Flag Set;
0018	24	UWORD serdatr;	serdatr:Serial Info;
001A	26	UWORD dskbytr;	dskbytr:Disk Info;
001C	28	UWORD intenar;	intenar:IntFlagSet;
001E	30	UWORD intreqr;	intreqr:IntFlagSet;
0020	32	APTR dskpt;	dskpt:ADDRESS;
0024	36	UWORD dsklen;	dsklen:CARDINAL;
0026	38	UWORD dskdat;	dskdat:CARDINAL;
0028	40	UWORD refptr;	refptr:CARDINAL;
002A	42	UWORD vposw;	vposw:LONGCARD;
002C	44	UWORD vhposw;	
002E	46	UWORD copcon;	copcon:BITSET;
0030	48	UWORD serdat;	serdat:Serial Info;
0032	50	UWORD serper;	serper:CARDINAL;
0034	52	UWORD potgo;	potgo:Pot Flag Set;
0036	54	UWORD joytest;	joytest:Coord;
0038	56	UWORD strequ;	strequ:CARDINAL;
003A	58	UWORD strvbl;	strvbl:CARDINAL;
003C	60	UWORD strhor;	strhor:CARDINAL;
003E	62	UWORD strlong;	strlong:CARDINAL;
0040	64	UWORD bltcon0;	bltcon0:BCØ Flag Set;
0042	66	UWORD bltcon1;	bltcon1:BC1 Flag Set;
0044	68	UWORD bltafwm;	bltafwm:BITSET;
0046	70	UWORD bltalwm;	bltalwm:BITSET;
0048	72	APTR bltcpt;	bltcpt:ADDRESS;
004B	76	APTR bltbpt;	bltbpt:ADDRESS;
0050	80	APTR bltapt;	bltapt:ADDRESS;
0054	84	APTR bltdpt;	bltdpt:ADDRESS;
0058	88	UWORD bltsize;	bltsize:CARDINAL;

7

Offset		C (hardware/custom.h)	Modula-2 (Hardware)
Hex	Dez		
005A	90	UWORD pad2d[3];	pad2d:ARRAY [0..2] OF CARDINAL;
0060	96	UWORD bltcmod;	bltcmod:CARDINAL;
0062	98	UWORD bltbmod;	bltbmod:CARDINAL;
0064	100	UWORD bltamod;	bltamod:CARDINAL;
0066	102	UWORD bltdmod;	bltdmod:CARDINAL;
0068	104	UWORD pad34[4];	pad34:ARRAY [0..3] OF CARDINAL;
0070	112	UWORD bltcdat;	bltcdat:CARDINAL;
0072	114	UWORD bltbdat;	bltbdat:CARDINAL;
0074	116	UWORD bltadat;	bltadat:CARDINAL;
0076	118	UWORD pad3b[4];	pad3b:ARRAY [0..3] OF CARDINAL;
007E	126	UWORD dsksync;	dsksync:CARDINAL;
0080	128	ULONG cop1lc;	cop1lc:ADDRESS;
0084	132	ULONG cop2lc;	cop2lc:ADDRESS;
0088	136	UWORD copjmp1;	copjmp1:CARDINAL;
008A	138	UWORD copjmp2;	copjmp2:CARDINAL;
008C	140	UWORD copins;	copins:CARDINAL;
008E	142	UWORD diwstrt;	diwstrt:Coord;
0090	144	UWORD diwstop;	diwstop:Coord;
0092	146	UWORD ddfstrt;	ddfstrt:Coord;
0094	148	UWORD ddfstop;	ddfstop:Coord;
0096	150	UWORD dmacon;	dmacon:DmaFlagSet;
0098	152	UWORD clxcon;	clxcon:Collision Control Flag Set;
009A	154	UWORD intena;	intena:IntFlagSet;
009C	156	UWORD intreq;	intreq:IntFlagSet;
009E	158	UWORD adkcon;	adkcon:AdkFlagSet;
00A0	160	struct AudChannel {	aud:Audio Channels (* RECORD
0000	0	UWORD *ac_ptr;	acptr:ADDRESS;
0004	4	UWORD ac_len;	aclen:CARDINAL;
0006	6	UWORD ac_per;	acper:CARDINAL;
0008	8	UWORD ac_vol;	acvol:CARDINAL;
000A	10	UWORD ac_dat;	acdat:CARDINAL;
000C	12	UWORD ac_pad[2];	acpad:ARRAY [0..1] OF CARDINAL;
0010	16	} aud[4];	END; *)
00E0	224	APTR bplpt[6];	bplpt:ARRAY [0..5] OF ADDRESS;
00F8	248	UWORD pad7c[4];	pad7c:ARRAY [0..3] OF CARDINAL;
0100	256	UWORD bplcon0;	bplcon0:BPLCØ Flag Set;

Offset		C (hardware/custom.h)	Modula-2 (Hardware)
Hex	Dez		
0102	258	UWORD bplcon1;	bplcon1:CARDINAL;
0104	260	UWORD bplcon2;	bplcon2:CARDINAL;
0106	262	UWORD pad83;	pad83:CARDINAL;
0108	264	UWORD bpl1mod;	bpl1mod:CARDINAL;
010A	266	UWORD bpl2mod;	bpl2mod:CARDINAL;
010C	268	UWORD pad86[2];	pad86:ARRAY [0..1] OF CARDINAL;
0110	272	UWORD bpldat[6];	bpldat:ARRAY [0..5] OF CARDINAL;
011C	284	UWORD pad8e[2];	pad8e:ARRAY [0..1] OF CARDINAL;
0120	288	APTR sprpt[8];	sprpt:ARRAY [0..7] OF ADDRESS;
0142	320	struct SpriteDef	spr:Sprites
		{	(* RECORD
0000	0	UWORD pos;	pos:CARDINAL;
0002	2	UWORD ctl;	ctl:CARDINAL;
0004	4	UWORD dataa;	data:LONGCARD;
0006	6	UWORD datab;	datab:CARDINAL;
0008	8	} spr[8];	END; *)
0180	384	UWORD color[32];	color:ARRAY [0..31] OF CARDINAL;
01C0	448	};	END;

Assembler Es sind nur die einzelnen Offsets definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Die Custom-Datenstruktur stellt nichts anderes als eine Aufzählung der Amiga-Hardware-Register auf den drei Spezialchips »Agnus«, »Denise« und »Paula« dar. Die Basisadresse des Registersatzes ist gleich 0x00DFF000 = 14675968. Die tatsächliche Registeradresse erhält man dementsprechend durch Addition des jeweiligen Offsets zur Basisadresse 0x00DFF000. In Modula-2 sind solche Umrechnungen allerdings überflüssig, da hier im Modul Hardware bereits eine Variable custom vom Typ Custom ab der absoluten Adresse 0x00DFF00 deklariert ist. In »C« umgeht man die Rechnerei durch eine kleine Zeigermanipulation:

```
struct Custom *custom;
.....
custom = (Custom)* 0x00DFF000;      /*  Zeiger auf Registersatz richten   */
```

Die nachfolgenden Erläuterungen zu den einzelnen Hardware-Registern sind alle einheitlich aufgebaut. Zuerst wird der (englische) Originalname des Registers angegeben. Danach finden Sie Informationen

bezüglich Lage und Zugriffsmodus des Registers, gefolgt von der eigentlichen Funktionserklärung. Die Bedeutungen der zur Lage- und Zugriffsmodus-Beschreibung verwendeten Abkürzungen sind leicht verständlich:

A: »Agnus«
D: »Denise«
P: »Paula«

ER: »early read« (DMA-Datentransfer ins RAM, der vor dem Lesezyklus des Prozessors stattfindet)

R: »read access« (Lesezugriff)

SW: »strobe write« (Schreibzugriff auf eine Strobe-Adresse ohne echte Registerbits)

W: »write access« (Schreibzugriff): Registerinhalt kann vom Copper nicht manipuliert werden.

*: Registerinhalt kann vom Copper nur dann verändert werden, wenn das CDANG-Bit = Bit 1 von copcon gesetzt ist.

Zwei allgemeine Hinweise bezüglich der Hardware-Register sind an dieser Stelle noch angebracht:

(1) Der Amiga besitzt viele durch Namen der Form xxx bzw. xxxw und xxxr gekennzeichnete Registerpaare. Beide Register betreffen die gleiche Hardware-Funktion, wobei die xxxr-Register reine Statusregister mit ausschließlichem Lesezugriff und die xxxRegister die eigentlichen Setzregister mit ausschließlichem Schreibzugriff sind. Im Rahmen der Funktionsbeschreibung solcher Registerpaare wird deshalb häufig auf das korrespondierende Schwesterregister verwiesen.

(2) Die in Custom definierten DMA-Adreßzeiger haben momentan lediglich eine Adreßbreite von 19 Bit, was gerade den 512K = 2^{19}-Bit-Chip-RAM entspricht. Da von diesen 19 Bit das niederwertigste unveränderbar gleich null gesetzt ist, arbeiten die DMA-Operationen alle nur auf Wortgrenzen. Lediglich die Bits 1 bis 18 der DMA-Adreßzeiger sind somit überhaupt von Bedeutung und manipulierbar.

❑ **bltdDat 0x0000 (0)**

Name: »blitter destination data«
Chip: A, Zugriff: ER -
Funktion: Zwischenspeicherung der Zieldaten einer Blitter-Operation, bis sie tatsächlich ins RAM geschrieben werden. Auf die angegebene Dummy-Adresse kann der Prozessor nicht zugreifen.

❑ **dmaconr 0x0002 (2)**

Name: »DMA control (and blitter status) read«
Chip: A P, Zugriff: R -
Funktion: Statusregister zur Kontrolle sämtlicher DMA-Kanäle des Amiga und der DMA-Statusbits des Blitters. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	SET/CLR	»set/clear control« (R/W) legt fest, ob das Schreiben einer 1 in ein Bit des Registers dieses löscht oder setzt.
14	BBUSY	»blitter busy« (R): Ist der Blitter gerade in Aktion?
13	BZERO	»blitter zero« (R): Ergab letzte Blitter-Operation eine Null?
12	-/-11	-/-
10	BLTPRI	»blitter DMA priority« (R/W): Hat Blitter Priorität vor CPU?
09	DMAEN	»DMA enable« (R/W): Ist überhaupt DMA eingeschaltet?
08	BPLEN	»bitplane enable« (R/W): Ist die Bitplane-DMA eingeschaltet?
07	COPEN	»copper enable« (R/W): Ist die Copper-DMA eingeschaltet?
06	BLTEN	»blitter enable« (R/W): Ist die Blitter-DMA eingeschaltet?
05	SPREN	»sprite enable« (R/W): Ist die Sprite-DMA eingeschaltet?
04	DSKEN	»disk enable« (R/W): Ist die Disk-DMA eingeschaltet?
03	AUD3EN	»audio 3 enable« (R/W): Ist Audio-DMA Kanal 3 eingeschaltet?
02	AUD2EN	»audio 2 enable« (R/W): Ist Audio-DMA Kanal 2 eingeschaltet?
01	AUD1EN	»audio 1 enable« (R/W): Ist Audio-DMA Kanal 1 eingeschaltet?
00	AUD0EN	»audio 0 enable« (R/W): Ist Audio-DMA Kanal 0 eingeschaltet?

- **vposr 0x0004 (4)**
  Name: »vertical position (most significant bit & frame flop) read«
  Chip: A, Zugriff: R -
  Funktion: Statusregister für die vertikale Position des Elektronenstrahls oder eines Lightpens. Das Bit 0 dieses Registers stellt das Bit 8 (also V8) des Werts der vertikalen Position des Elektronenstrahls dar (siehe *vhposr*). Weiterhin ist dem Wert des Bits 15 *LOF* zu entnehmen, ob gerade ein »long frame« (*LOF* gesetzt) oder ein »short frame« (*LOF* gelöscht) dargestellt wird.

- **vhposr 0x0006 (6)**
  Name: »vertical and horizontal position read«
  Chip: A, Zugriff: R -
  Funktion: Statusregister für vertikale und horizontale Position des Beams oder eines Lightpens. Die Bits 0 bis 7 stellen die niederwertigen Bits H0-H7 der horizontalen Position dar, die Bits 8 bis 15 entsprechend die niederwertigen Bits V0-V7 der vertikalen Position. Die Auflösung beträgt dabei 280 ns oder 1/160 der Monitorbreite.

- **dskdatr 0x0008 (8)**
  Name: »disk DMA data read«
  Chip: P, Zugriff: ER -
  Funktion: Dies ist der 2 Byte große Disk-DMA-Datenpuffer für gelesene oder noch zu schreibende Daten.

- **joy0dat 0x000A (10)**
  Name: »joystick/mouse 0 data«
  Chip: D, Zugriff: R -
  Funktion: Statusregister für die gesampelten Positionswerte des linken Maus- bzw. Joystick-Ports.

❑ **joy1dat 0x000C (12)**

Name: »joystick/mouse 1 data«
Chip: D, Zugriff: R -
Funktion: Statusregister für die gesampelten Positionswerte des rechten Maus- bzw. Joystick-Ports.

❑ **clxdat 0x000E (14)**

Name: »collision data«
Chip: D, Zugriff: R -
Funktion: Statusregister der von der Hardware erkannten Kollisionen, dessen Inhalt bei jedem Lesezugriff automatisch auf null zurückgesetzt wird. Für die einzelnen Bits des Registers gelten folgende Zuordnungen (siehe auch clxcon):

Bit	Bedeutung
15	-/-
14	Sprite 4 (oder 5) mit Sprite 6 (oder 7) kollidiert?
13	Sprite 2 (oder 3) mit Sprite 6 (oder 7) kollidiert?
12	Sprite 2 (oder 3) mit Sprite 4 (oder 5) kollidiert?
11	Sprite 0 (oder 1) mit Sprite 6 (oder 7) kollidiert?
10	Sprite 0 (oder 1) mit Sprite 4 (oder 5) kollidiert?
09	Sprite 0 (oder 1) mit Sprite 2 (oder 3) kollidiert?
08	Playfield 2 und Sprite 6 (oder 7) kollidiert?
07	Playfield 2 und Sprite 4 (oder 5) kollidiert?
06	Playfield 2 und Sprite 2 (oder 3) kollidiert?
05	Playfield 2 und Sprite 0 (oder 1) kollidiert?
04	Playfield 1 und Sprite 6 (oder 7) kollidiert?
03	Playfield 1 und Sprite 4 (oder 5) kollidiert?
02	Playfield 1 und Sprite 2 (oder 3) kollidiert?
01	Playfield 1 und Sprite 1 (oder 0) kollidiert?
00	Playfield 1 mit Playfield 2 kollidiert?

❑ **adkconr 0x0010 (16)**

Name: »audio and disk control read«
Chip: P, Zugriff: R *
Funktion: Statusregister zur Steuerung der Funktionsweise des Disk-Controllers und der vier Audio-Kanäle. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	SET/CLR	»set/clear control« (R/W) legt fest, ob das Schreiben einer 1 in ein Bit des Registers dieses löscht oder setzt.
14	PRECOMP1	Der binäre Wert der beiden PRECOMPx-Bits legt die Schreib-
13	PRECOMP0	vorkompensation für den Disk I/O fest: 00 = keine, 01 = 140 ns, 10 = 280 ns, 11 = 560 ns

Bit	Name	Bedeutung
12	MFMPREC	gesetzt = MFM Schreibvorkompensation, gelöscht = GCR Schreibvorkompensation
11	UARTBRK	Das Setzen dieses Bits löst einen UART-Interrupt aus.
10	WORDSYNC	Ist die Disketten-Lesesynchronisation mit dem disksync-Wert eingeschaltet?
09	MSBSYNC	Ist die Disketten-Lesesynchronisation für das Apple GCR- Format (höchstwertiges Bit 7 immer gesetzt!) eingeschaltet?
08	FAST	gesetzt = 2 us Taktzeit für Bit-Zellen auf Diskette (MFM), gelöscht = 4 us Taktzeit für Bit-Zellen auf Diskette (GCR)
07	USE3PN	Audio-Kanal 3 zur Frequenzmodulation von nichts?
06	USE2P3	Audio-Kanal 2 zur Frequenzmodulation von Audio-Kanal 3?
05	USE1P2	Audio-Kanal 1 zur Frequenzmodulation von Audio-Kanal 2?
04	USE0P1	Audio-Kanal 0 zur Frequenzmodulation von Audio-Kanal 1?
03	USE3VN	Wie USE3PN, aber für Lautstärkenodulation.
02	USE2V3	Wie USE2P3, aber für Lautstärkenodulation.
01	USE1V2	Wie USE1P2, aber für Lautstärkenodulation.
00	USE0V1	Wie USE0P1, aber für Lautstärkenodulation.

- **pot0dat 0x0012 (18)**

  Name: »potentiometer counter data left pair«

  Chip: P, Zugriff: R *

  Funktion: Dieses Zählregister enthält die aktuellen Potentiometerwerte des linken Maus- bzw. Joystick-Ports. Das niederwertige Byte stellt dabei die x-Koordinate, das höherwertige dagegen die y-Koordinate dar. Gestoppt wird der Horizontalzähler durch ein Signal an Pin 32 (*POT0X*) von Paula und der Vertikalzähler durch ein Signal an Pin 33 (*POT0Y*).

- **pot1dat 0x0014 (20)**

  Name: »potentiometer counter data right pair«

  Chip: P, Zugriff: R *

  Funktion: Dieses Zählregister enthält die aktuellen Potentiometerwerte des rechten Maus- bzw. Joystick-Ports. Das niederwertige Byte stellt dabei die x-Koordinate, das höherwertige dagegen die y-Koordinate dar. Gestoppt wird der Horizontalzähler durch ein Signal an Pin 35 (*POT1X*) von Paula und der Vertikalzähler durch ein Signal an Pin 36 (*POT1Y*).

- **potinp 0x0016 (22)**

  Name: »potentiometer port data read« (*POTGOR*)

  Chip: P, Zugriff: R *

  Funktion: Statusregister zur Kontrolle eines 4 Bit breiten bidirektionalen I/O-Ports, der die gleichen vier Pins von Paula benutzt wie die Potentiometerzähler pot0dat und pot1dat. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	OUTRY	gesetzt = Ausgabe an Pin 36, gelöscht = Eingabe an Pin 36
14	DATRY	Das gelesene bzw. auszugebende Datenbit für Pin 36.
13	OUTRX	gesetzt = Ausgabe an Pin 35, gelöscht = Eingabe an Pin 35
12	DATRX	Das gelesene bzw. auszugebende Datenbit für Pin 35.
11	OUTLY	gesetzt = Ausgabe an Pin 33, gelöscht = Eingabe an Pin 33
10	DATLY	Das gelesene bzw. auszugebende Datenbit für Pin 33.
09	OUTLX	gesetzt = Ausgabe an Pin 32, gelöscht = Eingabe an Pin 32
08	DATLX	Das gelesene bzw. auszugebende Datenbit für Pin 32.
07 .. 01	-/-	Ein Identifikationscode des Chips, der momentan gleich 0 ist.
00	START	Initialisiere und starte die Potentiometerzähler.

❑ **serdatr 0x0018 (24)**

Name: »serial port data and status read«

Chip: P, Zugriff: R *

Funktion: Aus diesem Register liest man die Datenbits vom Empfangspuffer der seriellen Schnittstelle sowie deren aktuellen Status ein. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

7

Bit	Name	Bedeutung
15	OVRUN	Ist serielle Schnittstelle beim Empfang nicht mehr mitgekommen?
14	RBF	Ist der Empfangspuffer der seriellen Schnittstelle voll?
13	TBE	Ist der Sendepuffer der seriellen Schnittstelle leer?
12	TSRE	Ist das Sende-Schieberegister der seriellen Schnittstelle leer?
11	RXD	Empfängt der RXD-Pin serielle UART-Daten, die vom Prozessor direkt getestet werden müssen?
10	-/-	
09	STP	Stopbit
08	DB8	Datenbit oder Stopbit (falls lang)
07 .. 00	DB7 ... DB0	Insgesamt acht Datenbits DB7 bis DB0.

❑ **dskbytr 0x001A (26)**

Name: »disk data byte and status read«

Chip: P, Zugriff: R *

Funktion: Datenpuffer und Statusregister für den Datentransfer zwischen Diskette und RAM durch den Disketten-Controller. Der Datentransfer erfolgt dabei byteorientiert, immer schön ein Byte nach dem anderen. Das »ready«-Statusbit 15 wird nach jedem Lesezugriff auf das Register automatisch zurückgesetzt. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	DSKBYT	Kann das nächste übertragene Datenbyte gelesen werden?
14	DMAON	Logische UND-Verknüpfung des DMAEN-Bits = Bit 15 von dsklen mit dem DMAEN-Bit = Bit 9 von dmacon.
13	DISKWRITE	Kopie des WRITE-Bits = Bit 14 von dsklen.
12	WORDEQUAL	Solange das von Diskette gelesene Datenbyte gleich dem Wert von dsksync ist, bleibt dieses Bit gesetzt.
11	-/-	
..	-/-	
08	-/-	
07		
..	DATA	Insgesamt 8 Bits für das eigentliche Datenbyte.
00		

❑ **intenar 0x001C (28)**

Name: »interrupt enable bits read«

Chip: P, Zugriff: R *

Funktion: Statusregister zur Verwaltung erlaubter und verbotener Interrupts. Erst das Setzen der entsprechenden Bits in dem zu diesem Statusregister gehörigen Setzregisters intena macht Interrupts der angegebenen Art möglich. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	SET/CLR	»set/clear control« (R/W) legt fest, ob das Schreiben einer 1 in ein Bit des Registers dieses löscht oder setzt.
14	INTEN	»master interrupt«: Sind Interrupts überhaupt erlaubt? Das Löschen dieses Bits desaktiviert sämtliche Interrupts.
13	EXTER	Externer Interrupt an Systempin INT6* liegt an.
12	DSKSYN	Wert in dsksyn ist gleich dskbytr-Datenbyte: DSKSYN-Interrupt.
11	RBF	Empfangspuffer der seriellen Schnittstelle voll: RBF-Interrupt
10	AUD3	Audio-Kanal 3 ist fertig: Audio-Interrupt Nr.3
09	AUD2	Audio-Kanal 2 ist fertig: Audio-Interrupt Nr.2
08	AUD1	Audio-Kanal 1 ist fertig: Audio-Interrupt Nr.1
07	AUD0	Audio-Kanal 0 ist fertig: Audio-Interrupt Nr.0
06	BLIT	Der Blitter ist fertig: Blitter-Interrupt.
05	VERTB	Beginn des »vertical blanking«-Intervalls: VERTB-Interrupt.
04	COPER	Ein vom Copper explizit ausgelöster Copper-Interrupt.
03	PORTS	Externer Interrupt von Ports oder Timer an Systempin INT2*.
02	SOFT	Ein von einem Programm ausgelöster Software-Interrupt.
01	DSKBLK	Die letzte Disk-DMA-Anfrage ist fertig: DSKBLK-Interrupt.
00	TBE	Sendepuffer der seriellen Schnittstelle leer: TBE-Interrupt.

❑ **intreqr 0x001E (30)**

Name: »interrupt request bits read«
Chip: P, Zugriff: R *
Funktion: Statusregister für Interrupt-Anforderungen, die von der CPU abgefragt werden (können). Nur wenn im intenar-Register die entsprechenden Interrupts aktiviert sind, ist die Auslösung des angeforderten Interrupts möglich. Für die einzelnen Bits des Registers gelten dieselben Zuordnungen wie beim »intenar«-Register.

❑ **dskpt 0x0020 (32)**

Name: »disk pointer«
Chip: A, Zugriff: W
Funktion: DMA-Adreßzeiger für die Disk-DMA, der vom Copper oder der CPU vor der DMA-Aktivierung korrekt initialisiert werden muß.

❑ **dskLen 0x0024 (36)**

Name: »disk length«
Chip: P, Zugriff: W
Funktion: Der Wert dieses Registers gibt die Länge des durch die Disk-DMA zu übertragenden Datenblocks in Datenworten an und enthält darüber hinaus noch zwei DMA-Kontrollbits. Bei der tatsächlichen Durchführung des DMA-Auftrags wird der Längenwert so lange dekrementiert, bis er gleich null ist. Daraufhin erfolgt dann ein *DSKBLK*-Interrupt (siehe intenar). Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	DMAEN	»disk DMA enable«: Ist Disk-DMA überhaupt zu aktivieren?
14	WRITE	gesetzt = Schreibmodus (Datentransfer vom RAM auf Diskette) gelöscht = Lesemodus (Datentransfer von Diskette ins RAM)
13 .. 00	LENGTH	Die Länge des durch Disk-DMA zu übertragenden Datenblocks in Datenworten.

❑ **dskdat 0x0026 (38)**

Name: »disk DMA data write«
Chip: P, Zugriff: W
Funktion: Siehe *dskdatar*.

❑ **refptr 0x0028 (40)**

Name: »refresh pointer«
Chip: A, Zugriff: (W)
Funktion: Adreßgenerator (in Form eines Zählers) für die dynamische RAM-Auffrischung (engl. »refresh«). Das Register ist nur aus Testgründen beschreibbar, es sollte jedoch niemals von der CPU beschrieben werden!

❑ **vposw 0x002A (42)**

Name: »vertical position (most significant bit&frame flop) write«
Chip: A, Zugriff: W
Funktion: Siehe *vposr*.

❑ **vhposw 0x002C (44)**

Name: »vertical and horizontal position write«
Chip: A, Zugriff: W
Funktion: Siehe *vhposr*.

❑ **copcon 0x002E (46)**

Name: »copper control register«
Chip: A, Zugriff: W
Funktion: Der Wert dieses 1-Bit-Registers entscheidet darüber, ob der Copper auf die mit * gekennzeichneten Blitter-Register zugreifen kann oder nicht. Für das einzige Bit des Registers gilt folgende Zuordnung:

Bit	Name	Bedeutung
01	CDANG	»copper danger mode«: Nur wenn dieses Bit gesetzt ist, kann der Copper die Blitter-Hardware direkt manipulieren.

serdat 0x0030 (48)

Name: »serial port data and stop bits write«
Chip: P, Zugriff: W
Funktion: Ausgaberegister für Datenbytes via serieller Schnittstelle (siehe auch serdatr). Die höherwertigen Bits 10 bis 15 des geschriebenen Werts sollten immer gleich null sein. Das explizit anzugebende Stopbit an Bitposition 9 oder 8 legt die tatsächliche Länge des (überlangen) Ausgabebytes fest. Die eigentlichen Datenbits befinden sich in den Bits 0 bis 7 (oder 8).

serper 0x0032 (50)

Name: »serial port period and control«
Chip: P, Zugriff: W
Funktion: Kontrollregister für die Sende- und Empfangs-Baudrate sowie die Länge der empfangenen Datenbytes der seriellen Schnittstelle. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	LONG	gesetzt = Empfang überlanger 9-Bit-Datenbyte gelöscht = Empfang normaler 8-Bit-Datenbyte
14 .. 00	RATE	Diese 15 Bit lange Ganzzahl gibt die Baudrate der Übertragung an: Baudrate = (RATE+1)*0.0002794 [Baud]

❑ **potgo 0x0034 (52)**

Name: »potentiometer port data write and start«
Chip: P, Zugriff: W
Funktion: Kontrollregister für die 4 Bit breiten bidirektionalen I/O-Ports von Paula. Die Bedeutung der einzelnen Bits des Registers sind unter *potinp* erklärt.

7

- **joytest 0x0036 (54)**
  Name: »mouse/joystick port counters test«
  Chip: D, Zugriff: W
  Funktion: Nach dem Schreibzugriff auf das Register werden alle vier Maus- bzw. Joystick-Positionszähler der Register *pot0dat* und *pot1dat* gleichzeitig mit dem hier eingetragenen Wert initialisiert.

- **strequ 0x0038 (56)**
  Name: »strobe for horizontal synchronization with VB and EQU«
  Chip: D, Zugriff: SW
  Funktion: Aktivierung der horizontalen Synchronisation der Zieladresse mit »vertical blanking« und EQU(???).

- **strvbl 0x003A (58)**
  Name: »strobe for horizontal synchronization with vertical blank«
  Chip: D, Zugriff: SW
  Funktion: Aktivierung der horizontalen Synchronisation der Zieladresse mit »vertical blanking«.

- **strhor 0x003C (60)**
  Name: »strobe for horizontal synchronization«
  Chip: D P, Zugriff: SW
  Funktion: Aktivierung der horizontalen Synchronisation Zieladresse.

7

- **strlong 0x003E (62)**
  Name: »strobe for identification of long horizontal line«
  Chip: D, Zugriff: SW
  Funktion: Identifikation langer Zeilen.

- **bltcon0 0x0040 (64)**
  Name: »blitter control register 0«Chip: A, Zugriff: W
  Funktion: Zusammen mit bltcon1 dient dieses Kontrollregister der vollständigen Steuerung des Blitters. Das LINE-Bit (Bit 0) von bltcon1 legt den grundsätzlichen Arbeitsmodus fest (Flächen oder Linien). Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung im »Flächenmodus« (LINE-Bit = 0)
15 .. 12	ASH	Versatz für Quelle A (ASH0 in Bit 12, ASH3 in Bit 15)
11	USEA	Soll Quelle A überhaupt benutzt werden?
10	USEB	Soll Quelle B überhaupt benutzt werden?
09	USEC	Soll Quelle C überhaupt benutzt werden?
08	USED	Soll Zielbereich D überhaupt benutzt werden?
07 .. 00	LF	Der Mintermwert zur Auswahl der während der Blitter-Operation vorzunehmenden logischen Verknüpfung der Quelldatenbits.

Bit	Name	Bedeutung im »Linienmodus« (LINE-Bit = 1)
15		
..	START	Horizontalversatz des Startpunkts der Linie
12		(START0 in Bit 12, START3 in Bit 15)
11	1	
10	0	
09	1	
08	1	
07		
..	LF	Der Mintermwert sollte mit 0x4A = 74 zur Auswahl der logischen
00		Verknüpfung D := AC+ABC initialisiert werden.

bltcon1 0x0042 (66)
Name: »blitter control register 1«
Chip: A, Zugriff: W
Funktion: Zusammen mit *bltcon0* dient dieses Kontrollregister der vollständigen Steuerung des Blitters. Das LINE-Bit (Bit 0) von bltcon1 legt den grundsätzlichen Arbeitsmodus fest (Flächen oder Linien). Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung im »Flächenmodus« (LINE-Bit = 0)
15		
..	BSH	Versatz für Quelle B (BSH0 in Bit 12, BSH3 in Bit 15)12
11	-/-	
..	-/-	
05	-/-	
04	EFE	Ist im exklusiven Füllmodus zu arbeiten?
03	IFE	Ist im inklusiven (normalen!) Füllmodus zu arbeiten?
02	FCI	Dieses Füllbit wird bei jeder 1 im Quellbereich getoggelt.
01	DESC	gesetzt = absteigender Adreßmodus (*bltxpt* werden dekrementiert)
		gelöscht = zunehmender Adreßmodus (*bltxpt* werden inkrementiert)
00	0	Flächenmodus eingeschaltet

Bit	Name	Bedeutung im »Linienmodus« (LINE-Bit = 1)
15		
..	TEXTURE	Versatz für Musterung der Linie, gemessen ab Startbit (TEXTURE0 in Bit 12, TEXTURE3 in Bit 15)
12		
11	0	
..	0	
07	0	

7

Bit	Name	Bedeutung im »Linienmodus« (LINE-Bit = 1)
15		
..	TEXTURE	Versatz für Musterung der Linie, gemessen ab Startbit (TEXTURE0 in Bit 12, TEXTURE3 in Bit 15)
12		
11	0	
..	0	
07	0	
06	SIGN	Vorzeichenflags
05	0	
04	SUD	»sometimes up or down«
03	SUL	»sometimes up or left«
02	AUL	»always up or left«
01	SING	»single bit line«: Ein besonderer Zeichenmodus für Linien, die für das automatische Flächenfüllen durch den Blitter benötigt werden. Diese Linien sind überall genau einen Pixel breit.
00	1	Linienmodus eingeschaltet

Zusatzbemerkung: Die 3 Bits AUL, SUL und SUD legen den Oktanten fürs Zeichnen im Linienmodus gemäß der folgenden Zuordnungstabelle fest:

SUD	SUL	AUL	Oktant
1	1	1	0
0	0	1	1
0	1	1	2
1	1	1	3
1	0	1	4
0	1	0	5
0	0	0	6
1	0	0	7

Der Linienmodus des Blitters: Im Linienmodus ändert sich die Bedeutung mehrerer Blitter-Register gemäß der folgenden Belegungstabelle:

Register	Bits	Bitname	Wert	Bedeutung im Linienmodus (LINE-Bit = 1)
bltcon0	15–12	START		Horizontalversatz für Linienstartpunkt
	11–08	USE	1011	(fürs Zeichnen von Linien vorgegeben!)
bltcon1	15–12	BSH		Versatz des Linienmusters
	5			Reserviert für den Systemgebrauch!
	04–02			Codewert zur Auswahl des Oktanten (s.o.)
	1	SING		Kontrollbit für den 1-Pixel-Linienmodus

Register	Bits	Bitname	Wert	Bedeutung im Linienmodus (LINE-Bit = 1)
	0	LINE	1	Aktivierungsbit für Linienmodus
bltadat	15–00	0x8000		Fürs Linienzeichnen benötigter Indexwert
bltbdat	15–00			Linienmuster fürs Zeichnen
bltsize	15–06			Länge der Linie in Pixeln (max. 1024)
	05–00	0x02		(fürs Zeichnen von Linien vorgegeben!)
bltamod	15–00	4Y-4X		(dto.)
bltbmod	15–00	4Y		(dto.)
bltbmod	15–00			Breite des Basis-Displayelements
bltbmod	15–00			Breite des Basis-Displayelements
bltapt	15–00	2Y-X		(fürs Zeichnen von Linien vorgegeben!)
bltapt	15–00			Adresse des Startpunkts der Linie
bltapt	15–00			Adresse des Startpunkts der Linie

(Hierbei stellt X die Breite und Y die Höhe des kleinsten Rechtecks dar, das die Linie gerade noch umfaßt!)

❑ **bltafwm 0x0044 (68)**

Name: »blitter first word mask for source A«
Chip: A, Zugriff: W
Funktion: Der Wert dieser Maske wird mit dem ersten Datenwort aus Quelle A durch ein logisches UND verknüpft. Auf diese Weise können beliebige Bits des ersten Datenworts der Quelle A bewußt ausgeblendet werden.

❑ **bltalwm 0x0046 (70)**

Name: »blitter last word mask for source A«
Chip: A, Zugriff: W
Funktion: Der Wert dieser Maske wird mit dem letzten Datenwort aus Quelle A durch ein logisches UND verknüpft. Auf diese Weise können beliebige Bits des letzten Datenworts der Quelle A bewußt ausgeblendet werden.

❑ **bltcpt 0x0048 (72)**

Name: »blitter pointer to source C«
Chip: A, Zugriff: W
Funktion: Das gleiche wie *bltcpt*, bezogen auf Quelle C (siehe auch bltcon1 zur Bedeutung des Registers im Linienmodus).

❑ **bltbpt 0x004B (76)**

Name: »blitter pointer to source B«
Chip: A, Zugriff: W
Funktion: DMA-Adreßzeiger für Quelle B bei Durchführung einer DMA-Operation des Blitters. Dieser Adreßzeiger muß vor dem Start der Blitter-Operation mit der Anfangsadresse des zu bearbeitenden Speicherbereichs initialisiert werden.

7

- **bltapt 0x0050 (80)**
  Name: »blitter pointer to source A«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcpt*, bezogen auf Quelle A (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **bltdpt 0x0054 (84)**
  Name: »blitter pointer to destination D«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcpt*, bezogen auf Zielbereich D (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **bltsize 0x0058 (88)**
  Name: »blitter start and size«
  Chip: A, Zugriff: W
  Funktion: Der Wert der Bits 15 bis 6 namens H9–H0 des Registers legen die vertikale Ausdehnung = Höhe der Blitter-Operation fest, der Wert der Bits 5 bis 0, genannt W5–W0, dagegen die horizontale Ausdehnung = Breite (siehe auch bltcon1 zur Bedeutung des Registers im Linienmodus). Da ein Schreibzugriff auf dieses Register den Blitter automatisch startet, sollten hier die Dimensionswerte erst dann eingetragen werden, wenn alle übrigen Blitter-Register korrekt initialisiert sind.

- **pad2d 0x005A (90)**
  Nicht belegte Adressen im Registerbereich.

- **bltcmod 0x0060 (96)**
  Name: »blitter modulo for source C«
  Chip: A, Zugriff: W
  Funktion: Moduloregister für Quelle C, dessen Wert am Ende jeder bearbeiteten Datenzeile der Quelle C zu ihrem Adreßzeiger *bltcpt* hinzugezählt wird (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **bltbmod 0x0062 (98)**
  Name: »blitter modulo for source B«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcmod*, bezogen auf Quelle B (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **bltamod 0x0064 (100)**
  Name: »blitter modulo for source A«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcmod*, bezogen auf Quelle A (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **bltdmod 0x0066 (102)**
  Name: »blitter modulo for destination D«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcmod*, bezogen auf den Zielbereich D (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **pad34 0x0068 (104)**
  Nicht belegte Adressen im Registerbereich.

- **bltcdat 0x0070 (112)**
  Name: »blitter source C data«
  Chip: A, Zugriff: W
  Funktion: Das Dateneingangsregister des Blitters für Quelle C, das üblicherweise durch die Blitter-DMA mit Datenworten versorgt wird. Allerdings besteht ebenfalls die Möglichkeit, dieses Register mittels der CPU zu beschreiben, um es mit einem speziellen Wert zu belegen.

- **bltbdat 0x0072 (114)**
  Name: »blitter source B data«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcdat*, bezogen auf Quelle B (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **bltadat 0x0074 (116)**
  Name: »blitter source A data«
  Chip: A, Zugriff: W
  Funktion: Das gleiche wie *bltcdat*, bezogen auf Quelle B (siehe auch *bltcon1* zur Bedeutung des Registers im Linienmodus).

- **pad3b 0x0076 (118)**
  Nicht belegte Adressen im Registerbereich.

7

- **dsksync 0x007E (126)**
  Name: »disk read synchronization«
  Chip: P, Zugriff: W
  Funktion: Der in diesem Register enthaltene Wert dient als Übereinstimmungscode zur Lesesynchronisation des Laufwerks (siehe auch Bit 10 von *adkcon*).

- **cop1lc 0x0080 (128)**
  Name: »copper first location register«
  Chip: A, Zugriff: W
  Funktion: In diesem Register steht die Adresse, die der Copper anspringt, falls der Strobe *copjmp1* beschrieben wird.

- **cop2lc 0x0084 (124)**
  Name: »copper second location register«
  Chip: A, Zugriff: W
  Funktion: In diesem Register steht die Adresse, die der Copper anspringt, falls der Strobe *copjmp2* beschrieben wird.

- **copjmp1 0x0088 (136)**
  Name: »copper restart at first location«
  Chip: A, Zugriff: SW
  Funktion: Ein Schreibzugriff auf diesen Strobe veranlaßt den Copper zu einem indirekten Sprung, wobei die Sprungadresse dem Register *cop1lc* entnommen wird. Da auch der Copper selbst den Strobe beschreiben kann, ermöglicht dies echte Sprünge durch den Copper.

- **copjmp2 0x008A (138)**
  Name: »copper restart at second location«
  Chip: A, Zugriff: SW
  Funktion: Das gleiche wie *copjmp1*, bezogen auf einen indirekten Sprung per *cop2lc*.

- **copins 0x008C (140)**
  Name: »copper instruction fetch identify«
  Chip: A, Zugriff: W
  Funktion: Eine Dummy-Adresse, die immer dann vom Copper erzeugt wird, wenn er selbst einen neuen Befehl in sein Instruktionsregister lädt.

- **diwstrt 0x008E (142)**
  Name: »display window start (upper left)«
  Chip: A, Zugriff: W
  Funktion: Dieses Register enthält die Position der linken oberen Ecke des sichtbaren Displaybereichs (auch Displayfenster genannt), wobei das hochwertige Byte mit Namen *VSTART* die y-Koordinate und das niederwertige Byte *HSTART* die x-Koordinate darstellt. Normalerweise ist in *diwstrt* der Wert 0x2C81 = 11393 zu finden.

- **diwstop 0x0090 (144)**
  Name: »display window stop (lower right)«
  Chip: A, Zugriff: W
  Funktion: Dieses Register enthält den Positionswert der rechten unteren Ecke des sichtbaren Displaybereichs (auch Displayfenster genannt), wobei das hochwertige Byte mit Namen *VSTOP* die y-Koordinate und das niederwertige Byte *HSTOP* die x-Koordinate darstellt. Normalerweise ist in *diwstop* der Wert 0xF4C1 = 62657 zu finden.

7

- **ddfstrt 0x0092 (146)**
  Name: »display data fetch start (horizontal position)«
  Chip: A, Zugriff: W
  Funktion: Dieses Register enthält den Positionswert des linken Startpunkts der nächsten Displayzeile. Die Auflösung beträgt dabei genau 4 Pixel, so daß ein Positionswert von 4 dem Pixel 16 entspricht. Im Low-Resolution-Modus ist für ddfstrt der Wert 0x0038 = 56 üblich, im High-Resolution-Modus dagegen 0x003C = 60.

- **ddfstop 0x0094 (148)**
  Name: »display data fetch stop (horizontal position)«
  Chip: A, Zugriff: W
  Funktion: Dieses Register enthält den Positionswert des rechten Endpunkts der nächsten Displayzeile. Die Auflösung beträgt dabei genau 4 Pixel, so daß ein Positionswert von 100 dem Pixel 400 entspricht. Im Low-Resolution-Modus ist für ddfstop der Wert 0x00D0 = 208 üblich, im High-Resolution-Modus dagegen 0x00D4 = 212. Zwischen den beiden Registern *ddfstrt* und *ddfstop* besteht folgender allgemeiner Zusammenhang:

ddfstrt = ddfstop – 8*(Wortanzahl – 1) (im Low-Resolution Modus)
ddfstrt = ddfstop – 4*(Wortanzahl – 2) (im High-Resolution Modus)

Mit Wortanzahl ist dabei die Anzahl von Displaydatenworten pro Zeile gemeint.

- ❑ **dmacon 0x0096 (150)**
  Name: »DMA control write (clear or set)«
  Chip: A D P, Zugriff: W
  Funktion: Kontrollregister zu *dmaconr*.

- ❑ **clxcon 0x0098 (152)**
  Name: »collision control«
  Chip: D, Zugriff: W
  Funktion: Die Werte der einzelnen Bits dieses Kontrollregisters entscheiden darüber, welche Bitplanes in welchem Zustand in die Kollisionserkennung mit einbezogen werden. Darüber hinaus steuern die vier hochwertigen Bits die Kollisionserkennung für ungerade Sprites. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	ENSP7	Ist Kollisionserkennung für Sprite 7 durchzuführen?
14	ENSP5	Ist Kollisionserkennung für Sprite 5 durchzuführen?
13	ENSP3	Ist Kollisionserkennung für Sprite 3 durchzuführen?
12	ENSP1	Ist Kollisionserkennung für Sprite 1 durchzuführen?
11	ENBP6	Ist Kollisionserkennung für Bitplane 6 durchzuführen?
10	ENBP5	Ist Kollisionserkennung für Bitplane 5 durchzuführen?
09	ENBP4	Ist Kollisionserkennung für Bitplane 4 durchzuführen?
08	ENBP3	Ist Kollisionserkennung für Bitplane 3 durchzuführen?
07	ENBP2	Ist Kollisionserkennung für Bitplane 2 durchzuführen?
06	ENBP1	Ist Kollisionserkennung für Bitplane 1 durchzuführen?
05	MVBP6	Match-Bit der Kollisionserkennung für Bitplane 6
04	MVBP5	Match-Bit der Kollisionserkennung für Bitplane 5
03	MVBP4	Match-Bit der Kollisionserkennung für Bitplane 4
02	MVBP3	Match-Bit der Kollisionserkennung für Bitplane 3
01	MVBP2	Match-Bit der Kollisionserkennung für Bitplane 2
00	MVBP1	Match-Bit der Kollisionserkennung für Bitplane 1

Beim Gebrauch der *ENSPx*-Bits ist zu beachten, daß die Sprites 0, 2, 4, 6 (mit geraden Nummern!) immer in die Kollisionserkennung mit einbezogen werden. Deshalb ergibt sich der endgültige Kollisionswert jeder Sprite-Gruppe aus einer logischen ODER-Verknüpfung der erkannten Kollisionen der zugehörigen geraden und ungeraden Sprites.

Der Wert der Match-Bits *MVBPx* legt die nötige »Polarität« der Bitkombination einer Kollision mit der entsprechenden Bitplane fest. Diese Polarität benötigt man, um Kollisionen nur mit Bereichen bestimmter Farben handhaben zu können. Wichtig ist außerdem, daß im Fall einer Desaktivierung aller ENBPx-Bits sämtliche Bitplane-Kollisionen registriert werden.

- ❑ **intena 0x009A (154)**
  Name: »interrupt enable bits (clear or set)«
  Chip: P, Zugriff: W
  Funktion: Kontrollregister zu Siehe *intenar*.

7

❑ **intreq 0x009C (156)**
Name: »interrupt request bits (clear or set)«
Chip: P, Zugriff: W
Funktion: Kontrollregister zu Siehe *intreqr*.

❑ **adkcon 0x009E (158)**
Name: »audio and disk control write«
Chip: P, Zugriff: W
Funktion: Kontrollregister zu Siehe *adkconr*.

❑ **aud 0x00A0 (160)**
Diese Komponente stellt kein echtes Hardware-Register dar. Vielmehr bezeichnet sie lediglich die Basisadresse eines Feldes von vier *AudChannel*-Datenstrukturen (Länge: 16 Byte), die die eigentlichen Register der einzelnen Audio-Kanäle 0 bis 3 enthalten. Die Offsets der nachfolgend aufgelisteten *AudChannel*-Komponenten gelten dabei immer relativ zur Basisadresse eines Audio-Kanals. Für Audio-Kanal k ergibt sich diese Basisadresse gemäß 0x00A0+k*0x0010 = 160+16*k.

❑ **>> ac_ptr ++ 0x0000 (0)**
Name: »audio channel x location« (AUDxLCH)
Chip: A, Zugriff: W
Funktion: DMA-Adreßzeiger auf den Speicherbereich im Chip-RAM, der die vom betreffenden Audio-Kanal abzuspielenden Audio-Daten enthält. Da dieser Zeiger beim Abspielen der Audio-Daten nicht verändert wird, muß er nur ein einziges Mal korrekt initialisiert werden.

7

❑ **>> ac_len ++ 0x0004 (4)**
Name: »audio channel x length« (AUDxLEN)
Chip: P, Zugriff: W
Funktion: Der Wert dieses Registers entspricht der Länge der vom betreffenden Audio-Kanal abzuspielenden Audio-Daten in Speicherworten.

❑ **>> ac_per ++ 0x0006 (6)**
Name: »audio channel x period« (AUDxPER)
Chip: P, Zugriff: W
Funktion: Der Registerwert legt die Frequenz (Rate) des DMA-Datentransfers für den betreffenden Audio-Kanal fest. Die minimale Rate beträgt hier 124 Taktzyklen, was einer maximalen Sampling-Frequenz von 28.86 kHz entspricht.

❑ **>> ac_vol ++ 0x0008 (8)**
Name: »audio channel x volume« (AUDxVOL)
Chip: P, Zugriff: W
Funktion: Der Lautstärkewert für den betreffenden Audio-Kanal, dessen Bits 0 bis 6 insgesamt 65 verschiedene, linear ansteigende Lautstärkestufen festlegen. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Bedeutung
15	
..	(nicht benutzt!)
07	
06	gesetzt = maximale Lautstärke (entspricht 64 Einsen ohne Nullen!)
05	
..	Dieser Wert legt eine von 64 Lautstärkestufen fest:
00	000000 = keine Ausgabe, 111111 = 63 Einsen und eine Null

- **>> ac_dat ++ 0x000A (10)**
  Name: »audio channel x data« (AUDxDAT)
  Chip: P, Zugriff: W
  Funktion: Der DMA-Datenpuffer des betreffenden Audio-Kanals, der die nächsten beiden, nacheinander auszugebenden Audio-Ausgabebytes im Zweierkomplement enthält. Normalerweise werden die Audio-Daten durch die Audio-DMA aus dem Chip-RAM in dieses Pufferregister geholt. Aber auch die CPU selbst kann hier Audio-Daten direkt ablegen.

- **>> ac_pad ++ 0x000C (12)**
  Nicht belegte Adressen im Registerbereich.

- **bplpt 0x00E0 (224)**
  Name: »bitplane x pointer« (BPLxPTH)
  Chip: A, Zugriff: W
  Funktion: Ein Feld von sechs DMA-Adreßzeigern für die Displaydaten der einzelnen Bitplanes. Da diese Zeiger beim Aufbau des Ausgabebildes durch die Displayhardware andauernd inkrementiert werden, müssen sie während jedes »vertical blanking« -Intervalls vom Copper oder der CPU neu initialisiert werden.

- **pad7c 0x00F8 (248)**
  Nicht belegte Adressen im Registerbereich.

- **bplcon0 0x0100 (256)**
  Name: »bitplane control register 0 (misc. control bits)«
  Chip: A D, Zugriff: W
  Funktion: Die drei *bplconx*-Register steuern die Benutzung und Funktionsweise der Bitplanes sowie anderer Aspekte der Display-Hardware. Dieses Register ist dabei für die Auflösung des Displays und eventuell angeschlossene Display-Peripherie zuständig. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	HIRES	Ist High-Resolution-Modus einzuschalten?
14		
..	BPU	»bitplane use code«: Dieser 3-Bit-Wert aus dem Bereich 000 bis 110
12		gibt die Anzahl der momentan zu benutzenden Bitplanes an.

7

Bit	Name	Bedeutung
11	HOMOD	Ist Hold-And-Modify-Modus einzuschalten?
10	DBPLF	Ist Dual-Playfield-Modus einzuschalten?
09	COLOR	Ist der Composite-Video Farbausgang zu aktivieren?
08	GAUD	»genlock audio enable«: Ist Audio-Genlock zu aktivieren? (Dann wird dieses während des »vertical blanking« -Intervalls auf dem *BKGND*-Pin gemultiplext.)
07	-/-	
..	-/-	
04	-/-	
03	LPEN	Ist der angeschlossene Lightpen zu aktivieren?
02	LACE	Ist Interlace-Modus einzuschalten?
01	ERSY	Ist Resynchronisation extern durchzuführen? (Dabei werden die *HSYNC*- und *VSYNC*-Pads zu Eingabekanälen.)
00	-/-	

❑ **bplcon1 0x0102 (258)**

Name: »bitplane control register 1 (horizontal scroll control)«
Chip: D, Zugriff: W
Funktion: Dieses Register ist für die horizontalen Scroll-Werte der Playfields zuständig (siehe auch bplcon0). Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	-/-	
..	-/-	
08	-/-	
07		
..	PF2H	Dieser 4 Bit umfassende Wert (Bit 7 = *PF2H3*, *PF2H0* = Bit 4)
04		legt den horizontalen Scroll-Code für Playfield 2 fest.
03		
..	PF1H	Dieser 4 Bit umfassende Wert (Bit 3 = *PF1H3*, *PF1H0* = Bit 0)
00		legt den horizontalen Scroll-Code für Playfield 1 fest.

❑ **bplcon2 0x0104 (260)**

Name: »bitplane control register 2 (video priority control)«
Chip: D, Zugriff: W
Funktion: Das letzte der drei bplconx-Register dient der Kontrolle der Video-Prioritäten der einzelnen Displayelemente untereinander. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15	-/-	
..	-/-	
07	-/-	
06	PF2PRI	gesetzt = Playfield 2 erscheint vor Playfield 1, gelöscht = Playfield 1 erscheint vor Playfield 2
05		
..	PF2P	Dieser Wert aus dem Bereich 000 bis 100 legt die relativen
03		Video-Prioritäten zwischen Playfield 2 und Sprite-Gruppen fest.
02		
..	PF1P	Dieser Wert aus dem Bereich 000 bis 100 legt die relativen
00		Video-Prioritäten zwischen Playfield 1 und Sprite-Gruppen fest.

Die *PFxP*-Werte der obigen Aufstellung haben dabei folgende Wirkung auf die Video-Prioritäten zwischen Playfield x (genannt *PFx*) und den vier Spritegruppen 0-1, 2-3, 4-5 und 6-7 (genannt *SPyz*):

PFxP	Lage des Playfields x und der Sprite-Gruppen zueinander
000	PFx > SP01 > SP23 > SP45 > SP67001 SP01 > PFx > SP23 > SP45 > SP67
010	SP01 > SP23 > PFx > SP45 > SP67
011	SP01 > SP23 > SP45 > PFx > SP67
100	SP01 > SP23 > SP45 > SP67 > PFx

❑ **pad83 0x0106 (262)**
Nicht belegte Adressen im Registerbereich.

❑ **bpl1mod 0x0108 (264)**
Name: »bitplane modulo 1 (odd planes)«
Chip: A, Zugriff: W
Funktion: Dieses Register enthält den Modulo-Wert aller Bitplanes mit ungeraden Nummern. Ein Modulo-Wert ist dabei nichts anderes als der Offset in Speicherworten zwischen dem Ende einer Zeile und dem Anfang der nächsten innerhalb des Displayspeichers.

❑ **bpl2mod 0x010A (266)**
Name: »bitplane modulo 2 (even planes)«
Chip: A, Zugriff: W
Funktion: Der in diesem Register enthaltene Modulo-Wert gilt nur für die Bitplanes mit geraden Nummern (siehe auch bpl1mod). Da die geraden und ungeraden Bitplanes eigene Modulo-Register besitzen, können ihre Größen durchaus unterschiedlich sein (und auch von der Größe des Displayausschnitts abweichen!).

❑ **pad86 0x010C (268)**
Nicht belegte Adressen im Registerbereich.

❑ **bpldat 0x0110 (272)**

Name: »bitplane x data (parallel-to-serial convert)« (*BPLxDAT*)
Chip: D, Zugriff: W
Funktion: Ein Feld von sechs Pufferregistern, in die üblicherweise von der Bitplane-DMA die nächsten, aus dem Chip-RAM geholten Ausgabedaten hineingeschrieben werden. Aber auch die CPU selbst kann diese Register direkt beschreiben.

❑ **pad8e 0x011C (284)**

Nicht belegte Adressen im Registerbereich.

❑ **sprpt 0x0120 (288)**

Name: »sprite x pointer« (*SPRxPTH*)
Chip: A, Zugriff: W
Funktion: Ein Feld von acht DMA-Adreßzeigern für die Displaydaten der Sprite-DMA. Diese Adreßzeiger müssen während jedes »vertical blanking«-Intervalls vom Copper oder der CPU wieder korrekt initialisiert werden.

❑ **spr 0x0142 (320)**

Diese Komponente stellt kein echtes Hardware-Register dar. Vielmehr bezeichnet sie lediglich die Basisadresse eines Feldes von acht Sprite-Def-Datenstrukturen (Länge: 8 Byte), die die eigentlichen Register der einzelnen Sprite-Kanäle 0 bis 7 enthalten.

❑ **>> pos ++ 0x0000 (0)**

Name: »sprite x vert.-horiz. start position data« (*SPRxPOS*)
Chip: A D, Zugriff: W
Funktion: Dieses Positionsregister enthält Teile der Koordinaten der Position des betreffenden Sprites. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15 .. 08	VSTARTL	Diese 8 Bit stellen das niederwertige Byte der y-Koordinate der Position der linken oberen Ecke des betreffenden Sprites dar.
07 .. 00	HSTARTH	Diese 8 Bit stellen das höherwertige Byte der x-Koordinate der Position der linken oberen Ecke des betreffenden Sprites dar.

❑ **>> ctl ++ 0x0002 (2)**

Name: »sprite x vert. stop position and control data« (*SPRxCTL*)
Chip: A D, Zugriff: W
Funktion: Das ist das eigentliche Kontrollregister des betreffenden Sprite-Kanals, das neben den fehlenden *HSTART*- und *VSTART*-Bits (siehe auch pos) noch den vertikalen Positionswert der rechten unteren Ecke des Sprites sowie das *ATT*-Bit zur Steuerung des Sprite-Attachements enthält. Für die einzelnen Bits des Registers gelten folgende Zuordnungen:

Bit	Name	Bedeutung
15		Diese 8 Bit stellen das niederwertige Byte der y-Koordinate
..	VSTOPL	der Position der rechten unteren Ecke des betreffenden Sprites
08		dar.
07	ATT	Gilt für diesen Sprite-Kanal ein Sprite-Attachement mit dem vorangehenden (nur für ungerade Sprite-Kanäle relevant)?
06	-/-	
..	-/-	
03	-/-	
02	VSTARTH	Das ist das hochwertige Bit 8 der *VSTART*-Koordinate (siehe pos).
01	VSTOPH	Das ist das hochwertige Bit 8 der *VSTOP*-Koordinate.
00	HSTARTL	Das ist das niederwertige Bit 0 der *HSTART*-Koordinate (siehe pos).

Die tatsächliche Höhe des dem betreffenden Sprite-Kanal zugeordneten Sprites in Low-Resolution Pixeln ergibt sich gemäß *VSTART-VSTOP*+1.

❑ **>> dataa ++ 0x0004 (4)**

Name: »sprite x image data register A« (*SPRxDATA*)

Chip: D, Zugriff: W

Funktion: In diesem DMA-Pufferregister werden üblicherweise von der Sprite-DMA aus dem Chip-RAM gelesene Image-Daten abgelegt, bevor sie von der Display-Hardware auf den Monitor gebracht werden. Aber auch die CPU selbst kann dieses Register direkt beschreiben. Ein Schreibzugriff auf dataa (durch die CPU oder den Copper) führt dabei zur Aktivierung des manuellen Sprite-Modus, in dem die Image-Daten aus den datax-Registern in jeder Zeile an der *HSTART*-Position ausgegeben werden. Durch einen nachfolgenden Schreibzugriff auf ctl können Sie den manuellen Sprite-Modus wieder desaktivieren.

❑ **>> datab ++ 0x0006 (6)**

Name: »sprite x image data register B« (*SPRxDATB*)

Chip: D, Zugriff: W

Funktion: Wie *dataa*, allerdings hat ein Schreibzugriff auf dieses Register keinen Einfluß auf den Sprite-Modus.

❑ **color 0x0180 (384)**

Name: »color table x« (*COLORxx*)

Chip: D, Zugriff: W

Funktion: Dieses Feld von 32 (Farb-)Registern wird üblicherweise als Farbpalette bezeichnet, aus der jeweils ein Farbregister durch den *bpldat*-Wert zur Pixeldarstellung ausgewählt wird. Die durch den Inhalt eines Farbregisters beschriebene Farbe setzt sich aus einer Mischung der drei Grundfarben Rot, Grün und Blau zusammen. Für jede Grundfarbe können 16 verschiedene Anteilswerte ausgewählt werden, so daß die in den Registern enthaltenen Farbwerte dem Format 0x0RGB entsprechen. (Hierbei stehen R, G, B als Platzhalter für Hexadezimalziffern!)

7

7.4 Die Resources-Library

Die Resources dienen zur Koordinierung des Zugriffs verschiedener Multitasking-Komponenten auf die Hardware.

DiscResource

Offset Hex	Offset Dez	C (resources/disc.h)	Modula-2 (Resources)
		struct DiscResource	DiscResource =
		{	RECORD
0000	0	struct Library dr_Library;	library:Library;
0022	34	struct DiscResourceUnit *dr_Current;	current:DiscResourceUnitPtr;
0026	38	UBYTE dr_Flags;	flags:DiscResourceFlagSet;
0027	39	UBYTE dr_pad;	pad:UByte;
0028	40	struct Library *dr_SysLib;	sysLib:LibraryPtr;
002C	44	struct Library *dr_CiaResource;	ciaResource:LibraryPtr;
0030	48	ULONG dr_UnitID[4];	unitID:ARRAY [0..3] OF LONGCARD;
0040	64	struct List dr_Waiting;	waiting:List;
004E	78	struct Interrupt dr_DiscBlock;	discBlock:Interrupt;
0064	100	struct Interrupt dr_DiscSync;	discSync:Interrupt;
007A	122	struct Interrupt dr_Index;	index:Interrupt;
0090	144	};	END;

Assembler — Präfix: *DR_*
Syntax: Definition der Parameter in Versalien. Der Parameter *dr_Library* ist nicht definiert.

Datenstrukturenreferenz — Keine.

Routinenreferenz — »Exec«: *AddResource*, *OpenResource*, *Rem Resource*.

»Resources«: *AllocUnit*, *FreeUnit*, *GetUnit*, *GetUnitId*, *GiveUnit*.

Beschreibung

Diese Datenstruktur dient dazu, den Zugriff verschiedener Tasks auf die Laufwerke zu koordinieren. Sie wird in die Resource-Liste der *ExecBase*-Datenstruktur aufgenommen.

- **dr_Library 0x0000 (0)** — Eine eingebundene *Library*-Datenstruktur, die die Resources-Library in das System einbindet.
- **dr_DiscResourceUnit 0x0022 (34)** — Ein Adreßzeiger auf die *DiscResourceUnit*-Datenstruktur des aktuellen Laufwerks.
- **dr_Flags 0x0026 (38)** — Folgende Flags geben an, welche Units als verfügbare Diskresources allokiert sind (in Modula-2 sind die Werte aus *DiscResourceFlagSet*):

C	Modula-2	Hex	Beschreibung
DRB_ALLOC0	*alloc0*	01	Unit 0 ist alloziert.
DRB_ALLOC1	*alloc1*	02	Unit 1 ist alloziert.
DRB_ALLOC2	*alloc2*	04	Unit 2 ist alloziert.
DRB_ALLOC3	*alloc3*	08	Unit 3 ist alloziert.
DRB_ACTIVE	*active*	10	Die aktuelle Unit ist gerade im Betrieb.

- **dr_Pad 0x0027 (39)** — Ein relativ unbedeutendes Füllbyte.
- **dr_SysLib 0x0028 (40)** — Manchmal ist dies ein Adreßzeiger auf die *ExecBase*-Datenstruktur.
- **dr_CiaResource 0x002C (44)** — ???
- **dr_UnitID 0x0030 (48)** — Ein Feld von vier LONG-Werten, in denen die ID's der Units 0 bis 3 stehen.
- **dr_Waiting 0x0040 (64)** — ???
- **dr_DiscBlock 0x004E (78)** — Eine eingebundene *Interrupt*-Datenstruktur, die den Interrupt-Server des Diskblock-Interrupts beschreibt.
- **dr_DiscSync 0x0064 (100)** — Eine eingebundene *Interrupt*-Datenstruktur, die den Interrupt-Server des Disksync (wird durch Finden der Sync-Markierung ausgelöst) beschreibt.
- **dr_Index 0x007A (122)** — Eine eingebundene *Interrupt*-Datenstruktur, die den Interrupt-Server des Diskindex (wird durch Finden der Index-Markierung ausgelöst) beschreibt.

DiscResourceUnit

Offset		C (resources/disc.h)	Modula-2 (Resources)
Hex	Dez	struct DiscResourceUnit	DiscResourceUnit =
		{	RECORD
0000	0	struct Message dru_Message;	message:Message;

Offset		C (resources/disc.h)	Modula-2 (Resources)
Hex	Dez		
0014	20	struct Interrupt dru_DiscBlock;	discBlock:Interrupt;
002A	42	struct Interrupt dru_DiscSync;	discSync:Interrupt;
0040	64	struct Interrupt dru_Index;	index:Interrupt
0056	86	};	END;

Assembler Syntax: *DiscResourceUnit —> DISCRESOURCEUNIT*

Der Parameter *dru_Message* ist nicht definiert.

Datenstrukturenreferenz »Resources«: *DiscResource.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur beschreibt ein einzelnes Laufwerk. Sie wird in die Liste der Units der *DiscResource*-Datenstruktur eingetragen.

- **dru_Message 0x0000 (0)** Eine eingebundene *Message*-Datenstruktur. So kann eine *DiscResourceUnit*-Datenstruktur an einen anderen Task geschickt werden.
- **dru_DiscBlock 0x0014 (20)** Eine eingebundene *Interrupt*-Datenstruktur, die den Interrupt-Server des Diskblock-Interrupts, beschreibt.
- **dru_DiscSync 0x002A (42)** Eine eingebundene *Interrupt*-Datenstruktur, die den Interrupt-Server des Disksync (wird durch Finden der Sync-Markierung ausgelöst) beschreibt.
- **dru_Index 0x0040 (64)** Eine eingebundene *Interrupt*-Datenstruktur, die den Interrupt-Server des Diskindex (wird durch Finden der Index-Markierung ausgelöst) beschreibt.

MiscResource

Offset		C (resources/misc.h)	Modula-2 (Resources)
Hex	Dez	struct MiscResource	MiscResource =
		{	RECORD
0000	0	struct Library mr_Library;	library: Library;
0022	34	ULONG mr_AllocArray[4];	allocArray: ARRAY [0..3] OF ADDRESS;
0032	50	};	END;

Assembler Der Parameter *mr_Library* ist nicht definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz »Resources«: *AllocMiscResource*, *FreeMiscResource*.

Beschreibung

Diese Datenstruktur dient dazu, den Zugriff des Multitasking auf eine beliebige Hardware-Resource zu koordinieren.

- **mr_Library 0x0000 (0)** Eine eingebundene *Library*-Datenstruktur, die die Resources-Library in das System einbindet.
- **mr_AllocArray 0x0022 (34)** Hier stehen Adreßzeiger auf die Resource-spezifischen Zusatzinformationen.

7.5 Die Workbench-Library

In dieser Library werden einige Datenstrukturen definiert, die die Verwaltung der Workbench unterstützen.

7

DiskObject

Offset		C (workbench/workbench.h)	Modula-2 (Workbench)
Hex	Dez	struct DiskObject	DiskObject =
		{	RECORD
0000	0	UWORD do_Magic;	magic:CARDINAL;
0002	2	UWORD do_Version;	version:CARDINAL;
0004	4	struct Gadget do_Gadget;	gadget:Gadget;
0030	48	UBYTE do_Type;	type:WBObjectType;
0032	50	char *do_DefaultTool;	defaultTool:ADDRESS;
0036	54	char ** do_ToolTypes;	toolTypes:ADDRESS;
003A	58	LONG do_CurrentX;	currentX:LONGINT;
003E	62	LONG do_CurrentY;	currentY:LONGINT;
0042	66	struct DrawerData *do_DrawerData;	drawerData:DrawerDataPtr;
0046	70	char *do_ToolWindow;	toolWindow:ADDRESS;
004A	74	LONG do_StackSize;	stackSize:LONGINT
004E	78	};	END;

Datenstrukturreferenz Keine.

Routinenreferenz »Icon«: *FreeDiskObject*, *GetDiskObject*, *GetIcon*, *PutDiskObject*, *PutIcon*.

Beschreibung

Die *DiskObject*-Datenstruktur wird von den »Icon«-Routinen zur Speicherung eines Icons auf externen Speichermedien in Form einer ».info«-Datei verwendet. Neben dem kompletten Icon-Gadget sind in der *DiskObject*-Datenstruktur noch das *Tooltype*-Feld, die Stackgröße, das »default tool« sowie die gewünschte Position des Icon-Gadgets auf dem Bildschirm und ggf. die Position und Ausmaße des zu öffnenden Verzeichnisfensters vorhanden.

❑ **do_Magic 0x0000 (0)**

Eine positive Ganzzahl, die der Identifikation einer extern gespeicherten Icon-Datei durch die »Icon«-Routinen *GetDiskObject* und *GetWBObject* dient. Gültige Icon-Dateien enthalten an dieser Stelle den Wert *WB_DISKMAGIC* (C) = 0xE310 = 58128 = *diskMagic* (Modula-2).

❑ **do_Version 0x0002 (2)**

Eine positive Ganzzahl, deren Wert die (Workbench-)Versionsnummer der Icon-Datei angibt. Um aufwärtskompatibel zu bleiben, sollte hier der Wert *WB_DISKVERSION* (C) = 0x0001 = 1 = *diskVersion* (Modula-2) eingetragen werden. Die »Icon«-Routinen setzen die richtige Versionsnummer automatisch!

❑ **do_Gadget 0x0004 (4)**

Eine vollständige *Gadget*-Datenstruktur (siehe »Intuition«), die das Icon-Gadget beschreibt, so wie es auf der Arbeitsfläche der Workbench erscheint. Bisher sind für Icon-Gadgets nur Boolean-Gadgets des Typs *GADGIMAGE* erlaubt, deren *PlanePick*-Wert gleich 3 sein muß und die lediglich *RELVERIFY* sowie *GADGIMMEDIATE* als Aktivierungsflags gesetzt haben sollten.

❑ **do_Type 0x0030 (48)**

Eine 2-Byte-Marke, deren einzelne Bits festlegen, um welchen Typ von Icon es sich tatsächlich handelt. Zur Benutzung sind die folgenden Werte vordefiniert:

C	Modula-2	Hex	Dez	Bedeutung
WBDISK	disk	0001	1	Es ist das Root-Verzeichnis einer Disk.
WBDRAWER	drawer	0002	2	Es ist ein Verzeichnis auf Diskette.
WBTOOL	tool	0003	3	Es ist ein direkt ausführbares Programm.
WBPROJECT	project	0004	4	Es ist irgendeine Datendatei.
WBGARBAGE	garbage	0005	5	Es ist das Trashcan-Verzeichnis.
WBDEVICE	device	0006	6	Es ist ein (logisches) Gerät.
WBKICK	kick	0007	7	Es ist keine Amiga-DOS-Diskette.

Wichtig: In Modula-2 ist hier der Aufzählungstyp *WBObjectType* zu benutzen.

❑ **do_DefaultTool 0x0032 (50)**

Ein Adreßzeiger auf eine C-Zeichenkette, die den Namen des Programms enthält, das beim Anklicken des Icons automatisch zur Bearbeitung der ausgewählten Datei gestartet werden soll. Dieser Adreßzeiger ist nur für Icons vom Typ *WBPROJECT* und *WBDISK* von Bedeutung! Bei *WBDISK*-

Icons steht hier der Name des Kopierprogramms, das benutzt wird, falls die Diskette als Quelldiskette einer Duplikationsoperation auftritt.

- **do_ToolTypes 0x0036 (54)** Ein Adreßzeiger auf ein Feld von Adreßzeigern auf C-Zeichenketten, deren Format frei gewählt werden kann. Dieses *Tooltype*-Feld dient der Übergabe von Status- und Umgebungsinformationen an ein Programm, das von der Workbench aus gestartet wurde.

- **do_CurrentX 0x003A (58)** Eine lange Ganzzahl, deren Wert die x-Koordinate festlegt, an der das Icon-Gadget auf dem Bildschirm erscheinen soll. Diese wird relativ zur linken oberen Ecke des Fensters gemessen, das dem Verzeichnis zugeordnet ist, welches das Icon enthält. Durch Angabe des Werts *NO_ICON_-POSITION* (C) = 0x80000000 = *MIN(LONGINT)* = *noIcon-Position* (Modula-2) sowohl in *do_CurrentX* als auch in *do_CurrentY* überläßt man die Aufgabe der Positionierung des Icon-Gadgets der Workbench.

- **do_CurrentY 0x003E (62)** Eine lange Ganzzahl, deren Wert die y-Koordinate festlegt, an der das Icon-Gadget auf dem Bildschirm erscheinen soll. Diese wird relativ zur linken oberen Ecke des Fensters gemessen, das dem Verzeichnis zugeordnet ist, welches das Icon enthält. Durch Angabe des Werts *NO_ICON_-POSITION* (C) = 0x80000000 = *MIN(LONGINT)* = *noIcon-Position* (Modula-2) sowohl in *do_CurrentX* als auch in *do_CurrentY* überläßt man die Aufgabe der Positionierung des Icon-Gadgets der Workbench.

- **do_DrawerData 0x0042 (66)** Ein Adreßzeiger auf eine *DrawerData*-Datenstruktur, die das (Verzeichnis-)Fenster beschreibt, das beim Anklicken dieses Icons zur Anzeige seines Inhalts geöffnet werden soll. Dieser Adreßzeiger ist nur für Icons vom Typ *WBDISK*, *WBDRAWER* und *WBGARBAGE* von Bedeutung, weil alle anderen keine Verzeichnisse darstellen.

- **do_ToolWindow 0x0046 (70)** Ein Adreßzeiger auf eine C-Zeichenkette, die ein *CON*:- oder *RAW*:-Fenster beschreibt, das beim Start des zu diesem Icon gehörigen Programms als Standardein-/Ausgabedatei geöffnet werden soll. Der Adreßzeiger auf die *Window*-Datenstruktur (siehe »Intuition«) des neuen Fensters wird dabei automatisch in die Komponente *pr_WindowPtr* der *Process*-Datenstruktur (siehe »Dos«) des Programms eingetragen. Um kein Fenster automatisch öffnen zu lassen, trägt man hier den Wert *NULL* ein. Dieser Adreßzeiger ist nur für Icons vom Typ *WBTOOL* von Bedeutung.

- **do_StackSize 0x004A (74)** Eine lange Ganzzahl, deren Wert die Größe des Stacks festlegt, der zur Ausführung des zu diesem Icon gehörigen

Programms verwendet werden soll. Ist der Wert von *do_-StackSize* gleich Null, so wird dem Programm ein Stack von vor einstellungsgemäß 4Kbyte zugeordnet. Dieser Adreßzeiger ist nur für Icons vom Typ *WBTOOL* von Bedeutung.

DrawerData

Offset		C (workbench/workbench.h)	Modula-2 (Workbench)
Hex	Dez	struct DrawerData	DrawerData =
		{	RECORD
0000	0	struct NewWindow dd_NewWindow;	newWindow:NewWindow;
0030	48	LONG dd_CurrentX;	currentX:LONGINT;
0034	52	LONG dd_CurrentY;	currentY:LONGINT;
0038	56	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Die *DrawerData*-Datenstruktur wird vom Workbench-Programm dazu benutzt, die Charakteristika des für ein Verzeichnis zu öffnenden Fensters zu speichern.

❑ **dd_NewWindow 0x0000 (0)** Eine *NewWindow*-Datenstruktur (siehe »Intuition«), deren Adresse an die »Intuition«-Routine *OpenWindow* zum Öffnen des Verzeichnis-Fensters übergeben wird.

❑ **dd_CurrentX 0x0030 (48)** Eine lange Ganzzahl, deren Wert die x-Koordinate der aktuellen Position der linken oberen Ecke des diesem Verzeichnis übergeordneten Verzeichnis-Fensters angibt.

❑ **dd_CurrentY 0x0034 (52)** Eine lange Ganzzahl, deren Wert die y-Koordinate der aktuellen Position der linken oberen Ecke des diesem Verzeichnis übergeordneten Verzeichnis-Fensters angibt.

FreeList

Offset		C (workbench/workbench.h)	Modula-2 (Workbench)
Hex	Dez	struct FreeList	FreeList =
		{	RECORD
0000	0	WORD fl_NumFree;	numFree:INTEGER;
0002	2	struct List fl_MemList;	memList:List
001A	16	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz »Icon«: *AddFreeList, FreeFreeList, GetIcon.*

Beschreibung

Die *FreeList*-Datenstruktur stellt eine einfache Speicherbereichsliste dar, die den »Icon«-Routinen zur Aufnahme der von ihnen dynamisch allokierten Speicherbereiche dient.

- **fl_NumFree 0x0000 (0)** Eine Ganzzahl, deren Wert die Anzahl freier Bytes in der ganzen Speicherbereichsliste angibt.

- **fl_MemList 0x0002 (2)** Eine *List*-Datenstruktur (siehe »Exec«), über die auf die einzelnen Elemente der Speicherbereichsliste zugegriffen werden kann. Die Speicherbereichsliste besteht dabei aus lauter *MemList*-Datenstrukturen (siehe »Exec«), in denen sämtliche allokierten Speicherbereiche eingetragen sind. Falls der von einer *FreeList*-Datenstruktur selbst belegte Speicherplatz in ihrer eigenen Speicherbereichsliste vermerkt werden soll, so muß dies im ersten Element der Liste geschehen!

7

WBArg

Offset		C (workbench/startup.h)	Modula-2 (Workbench)
Hex	Dez	struct WBArg	WBArg =
		{	RECORD
0000	0	BPTR wa_Lock;	lock:FileLockPtr;
0004	4	BYTE *wa_Name;	name:ADDRESS
0008	8	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Die *WBArg*-Datenstruktur wird vom Workbench-Programm in der *WBStartup*-Datenstruktur zur Übergabe der Argumente an neu gestartete Programme verwendet.

- **wa_Lock 0x0000 (0)** Ein Adreßzeiger auf eine *FileLock*-Datenstruktur (siehe »Dos«), die das Verzeichnis repräsentiert, in dem dieses Argument zu finden ist.

- **wa_Name 0x0004 (4)** Ein Adreßzeiger auf eine C-Zeichenkette, die den tatsächlichen Namen des Arguments enthält.

WBStartup

Offset		C (workbench/startup.h)	Modula-2 (Workbench)
Hex	Dez	struct WBStartup	WBStartup =
		{	RECORD
0000	0	struct Message sm_Message;	message:Message;
0014	20	struct MsgPort *sm_Process;	process:MsgPortPtr;
0018	24	BPTR sm_Segment;	segment:BPTR;
001C	28	LONG sm_NumArgs;	numArgs:LONGINT;
0020	32	char *sm_ToolWindow;	toolWindow:ADDRESS;
0024	36	struct WBArg *sm_ArgList;	argList:WBArgumentsPtr;
0028	40	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Die *WBStartup*-Datenstruktur stellt eine erweiterte »Exec«-Botschaft dar, die das Workbench-Programm als einleitende Startup-Information an jedes (durch Anklicken seines Icons!) neu gestartete Programm sendet.

7

❑ **sm_Message 0x0000 (0)**

Eine normale *Message*-Datenstruktur (siehe »Exec«), deren *mn_ReplyPort*-Komponente auf den Port des Workbench-Programms verweist. Diese Botschaft darf nur am Ende der Programmausführung mittels der »Exec«-Routine *ReplyMsg* beantwortet werden, weil das Workbench-Programm den von der Botschaft insgesamt belegten Speicherplatz dann wieder automatisch freigibt. Beantwortet ein Programm die Startup-Botschaft nicht, so hat es selbst für die korrekte Freigabe des belegten Speicherplatzes zu sorgen.

❑ **sm_Process 0x0014 (20)**

Ein Adreßzeiger auf die *Process*-Datenstruktur (siehe »Dos«), die für das neu gestartete Programm vom Workbench-Programm automatisch angelegt wird.

❑ **sm_Segment 0x0018 (24)**

Ein Adreßzeiger vom Typ BPTR auf die Segmentliste des geladenen Objektcodes des neu gestarteten Programms. Dieser Adreßzeiger ist das Ergebnis eines vom Workbench-Programm intern durchgeführten Aufrufs der »Exec«-Routine *LoadSeg*.

❑ **sm_NumArgs 0x001C (28)**

Eine lange Ganzzahl, deren Wert die Anzahl der Argumente angibt, die an das neu gestartete Programm übergeben wurden. Diese Zahl entspricht der Elementanzahl der Argumenttabelle *sm_ArgList*.

- **sm_ToolWindow 0x0020 (32)**

  Ein Adreßzeiger auf eine C-Zeichenkette, die eine Kopie der Zeichenkette *do_ToolWindow* aus der *DiskObject*-Datenstruktur des zum neu gestarteten Programm gehörigen Icons darstellt.

- **sm_ArgList 0x0024 (36)**

  Ein Adreßzeiger auf eine Tabelle von *WBArg*-Datenstrukturen, die die an das neu gestartete Programm übergebenen Argumente repräsentieren. Insgesamt enthält die Tabelle genau *sm_NumArgs* Argumenteinträge.

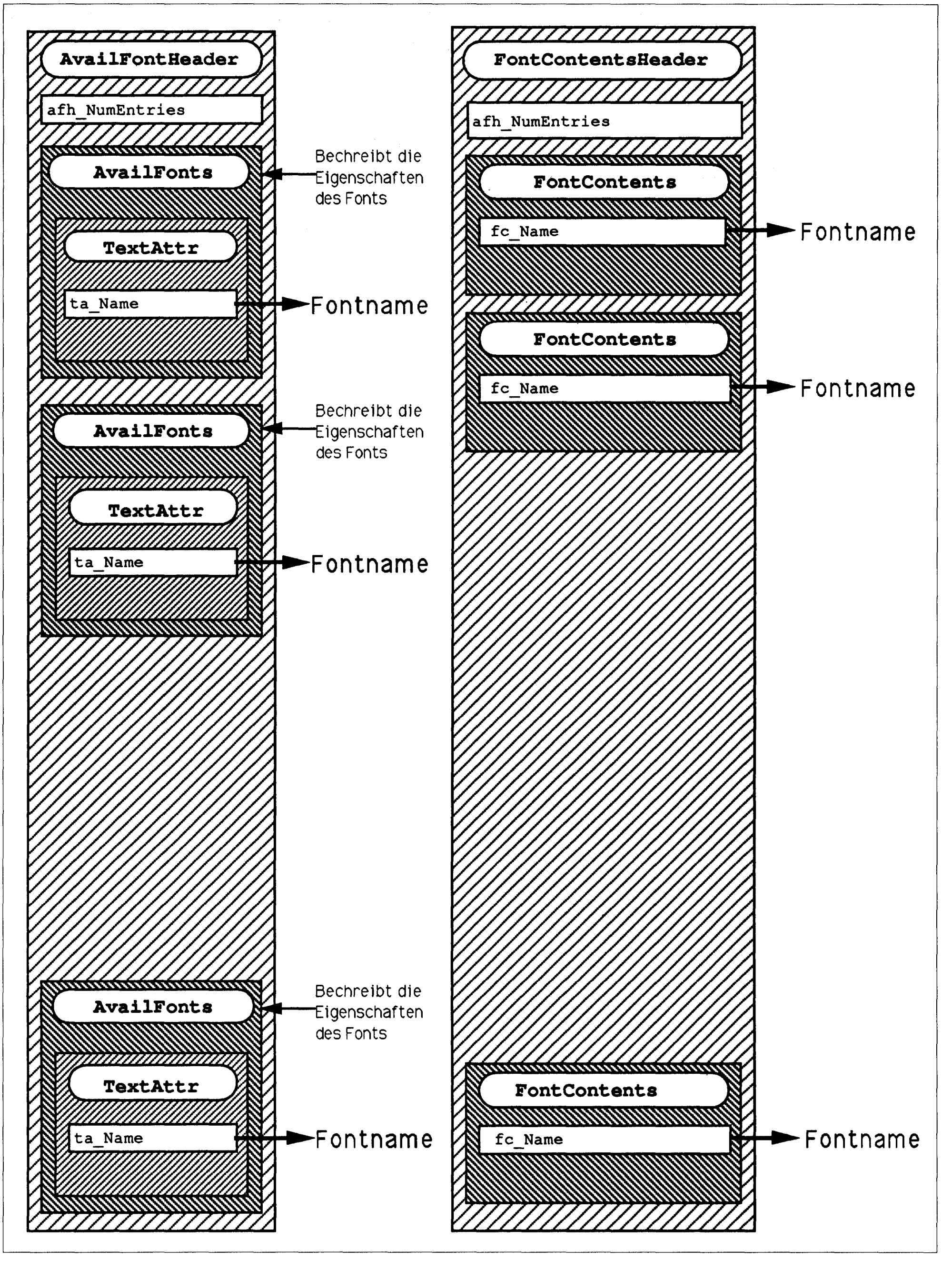


Abb. 7.1: Wichtige Datenstrukturen der Diskfont-Library

8 Die Devices

Bei den Devices handelt es sich in erster Linie um gerätespezifische Treiber. Ähnlich wie bei den Libraries sind in ihnen alle nötigen Routinen und Datenstrukturen definiert, die zur Programmierung eines spezifischen Ein- oder Ausgabegerätes nötig sind. Wie die Libraries müssen auch die Devices vor einem Zugriff durch ein Programm geöffnet und vor Beendigung desselben wieder geschlossen werden. Dazu gibt es die beiden Exec-Routinen *OpenDevice* und *CloseDevice*. Im Unterschied zu den Libraries läuft die Kommunikation mit den Devices im wesentlichen nur über eine *IOStdReq*-Datenstruktur, wobei sich diese von Device zu Device leicht verändern kann. Auf die Besonderheiten der zu einem Device gehörenden *IOStdReq*-Datenstruktur gehen wir jeweils am Anfang eines Abschnittes ein.

8.1 Das Audio-Device

Dieses Device dient, wie dem Namen schon zu entnehmen ist, zur Tonausgabe und gibt uns die vollständige Kontrolle über die Audio-Hardware des Amiga. Die Steuerung des Audio-Device erfolgt mit Hilfe der *IOAudio*-Datenstruktur. Geöffnet wird das Device mit Hilfe der OpenDevice-Routine, der als IORequest ein Adreßzeiger auf eine vorinitialisierte *IOAudio*-Datenstruktur übergeben werden muß. In *io_AllocKey* muß entweder null oder eine Kopie eines anderen gültigen AllocKeys stehen, in *io_AllocData* ein Adreßzeiger auf eine Kanalmaske und in *io_Length* die Anzahl der Bytes in der Kanalmaske. Das *io_Error*-Feld dieser *IOAudio*-Datenstruktur beinhaltet immer null, falls kein Fehler aufgetreten ist, und sonst den entsprechenden Fehlercode. *In io_Unit* steht vor der Ausführung eines Befehls die Maske der Soundkanäle, die von ihm betroffen werden sollen, und hinterher die der tatsächlich betroffenen. Dabei entsprechen die Bits 0–3 den Kanälen 1–4.

C	Modula-2	Dez	Beschreibung
CMD_RESET	reset	1	Alle Audioregister auf Defaultwerte rücksetzen.
CMD_READ	read	2	Einen Zeiger auf den aktuellen *WRITE*-Block (IO-Block der für den momentan stattfindenden Schreibzugriff zuständig ist) zurückgeben.

ioa_Data: Nach dem Aufruf ein Adreßzeiger auf den aktuellen *WRITE*-Block.

CMD_WRITE	write	3	Die Tonausgabe einleiten.

ioa_Data: Ein Adreßzeiger auf ein Feld (das im Chip-RAM liegen muß) von vorzeichenbehafteten Bytes, die die Wellenform festlegen.

ioa_Length: Die Länge des Feldes (aus *ioa_Data*) in Bytes (erlaubte Werte sind alle geraden Zahlen zwischen 2 und 131072).

ioa_Period: Die Länge des Tons als Vielfaches von 279,365 Nanosekunden (erlaubte Werte liegen zwischen 24 und 65536).
ioa_Volume: Die Lautstärke des Tones als ein Wert zwischen 0 und 64.
ioa_Cycles: Die Anzahl der Wiederholungen dieses Tones.
io_Flags: 0 = Ton ausgeben; *ADIOF_PERVOL* gesetzt = Lautstärke und Länge des Tons initialisieren; *ADIOF_WRITEMESSAGE* gesetzt = eine Write-Message beantworten.

C	Modula-2	Bit	Beschreibung
CMD_STOP	stop	6	Die Tonausgabe unterbrechen.
CMD_START	start	7	Die durch *STOP* unterbrochene Tonausgabe wieder aufnehmen.
CMD_FLUSH	flush	8	Alle *WRITE*-Befehle löschen.

Ihnen stehen für das Audio-Device folgende Befehle zusätzlich zur Verfügung (zu den Definitionen in C und Assembler gehört jeweils noch das Präfix *ADCMD_*):

FREE	free	09	Die Soundkanäle freigeben.
SETPREC	setPrec	10	Die Priorität setzen. Große Priorität bedeutet, daß der Kanal nicht weggenommen werden kann.

io_Node.ln_Pri: Die neue Priorität

8

FINISH	finish	11	Die Soundausgabe für Kanal (Kanäle) anhalten. Es wird *AbortIO* aufgerufen.

io_Flags: *IOF_QUICK* gesetzt = IORequest beantworten; *ADIOF_SYNCCYCLE* gesetzt = sofort stoppen; gelöscht = Ende des Zyklus abwarten.

PERVOL	perVol	12	Die Länge und die Lautstärke einstellen.

ioa_Period: Die neue Länge (siehe *CMD_WRITE*).
ioa_Volume: Die neue Lautstärke.

C	Modula-2	Bit	Beschreibung
LOCK	lock	13	Benachrichtigen, falls ein Kanal von einem anderen Task »weggenommen« wurde.

C	Modula-2	Bit	Beschreibung
WAITCYCLE	waitCycle	14	Auf das Ende des Ausgabezyklus warten.
ALLOCATE	allocate	32	Die Soundkanäle einrichten.

ioa_Data: Ein Adreßzeiger auf die zu allozierende Kanalmaske.
ioa_Length: Die Länge der Kanalmaske.
io_Flags: *IOF_QUICK* gelöscht = *IORequest* beantworten; gesetzt = nur asynchrone *IORequests* beantworten; *ADIOF_NOWAIT* gelöscht = warten, bis Allozierung erfolgreich abgeschlossen (und wenn es noch so lange dauert); gesetzt = sofort mit Fehlermeldung zurückkehren, falls Allokierung nicht erfolgreich;

Folgende Bits sind für *io_Flags* definiert (zu den C- und Assemblerversionen gehört jeweils das Präfix *ADIOF_*; in Modula-2 entsprechen sie den jeweiligen Bits aus *IOFlagSet*):

C	Modula-2	Bit	Beschreibung
PERVOL	perVol	4	Die Dauer und die Lautstärke bei *ACMD_ALLOCATE* setzen.
SYNCCYCLE	syncCycle	5	Vor Beenden der Ausgabe mit *CMD_-FINSH* soll der aktuelle Zyklus beendet werden.
NOWAIT	noWait	6	Auf Warten beim Bereitstellen mit *ACMD_ALLOCATE* verzichten.
WRITEMESSGAE	writeMessage	7	Es wird in ioa_WriteMessage mitgeteilt, wenn die Bearbeitung von *CMD_WRITE* aufgenommen wird.

Folgende Fehlermeldungen sind für das Audio-Device zusätzlich definiert (zu den Definitionen in C und Assembler gehört jeweils das Präfix *ADIOERR_*):

C	Modula-2	Dez	Beschreibung
NOALLOCATION	noAllocation	–10	Es wurde ein falscher AllocKey angegeben.
ALLOCFAILED	allocFailed	–11	Kanal konnte nicht bereitgestellt werden.
CHANNELSTOLEN	channelStolen	–12	Der Kanal wurde von einem anderen Task »weggenommen«.

IOAudio

Offset		C (devices/audio.h)	Modula-2 (Audio)
Hex	Dez	struct IOAudio	IOAudio =
		{	RECORD
0000	0	struct IORequest ioa_Request;	request:IORequest;
0020	32	WORD ioa_AllocKey;	allocKey:INTEGER;
0022	34	UBYTE *ioa_Data;	data:ADDRESS;
0026	38	ULONG ioa_Length;	length:LONGCARD;
002A	42	UWORD ioa_Period;	period:CARDINAL;
002C	44	UWORD ioa_Volume;	volume:CARDINAL;
002E	46	UWORD ioa_Cycles;	cycles:CARDINAL;
0030	48	struct Message ioa_WriteMsg;	writeMsg:Message;
0044	68	};	END;

Assembler Der Parameter *ioa_Request* ist nicht definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz »Exec«: *CloseDevice*, *OpenDevice*.

Beschreibung

Bei dieser Datenstruktur handelt es sich um eine für das Audio-Device erweiterte *IORequest*-Datenstruktur.

8

❑ **ioa_Request 0x0000 (0)**

Dies ist eine gewöhnliche *IORequest*-Struktur, wie sie im Kapitel über die Exec-Library beschrieben ist. Über die entsprechenden Felder dieser Struktur können Befehle an das Device geschickt und Fehlermeldungen gelesen werden. Das *ln_Pri-Feld* des io_MessageFeldes von *io_Request* wird benutzt, um die »Dringlichkeit« Ihres Sounds anzugeben. Ihnen stehen für das Audio-Device folgende Befehle standardmäßig zur Verfügung:

❑ **ioa_AllocKey 0x0020 (32)**

Beim Allokieren des ersten Kanals muß hier eine 0 übergeben werden, wodurch der *ACMD_ALLOCATE*-Befehl hier einen Key hineinschreibt, der später nicht verändert werden darf.

❑ **ioa_Data 0x0022 (34)**

Ein Zeiger auf die Daten, die dem Device übergeben werden, also beispielsweise die Kanalmaske beim *ACMD_ALLOCATE*-Befehl oder die Wellenform beim *CMD_WRITE*-Befehl. Diese Daten müssen sich im Chip-RAM befinden!

❑ **ioa_Length 0x0026 (38)**

Dieser Wert gibt die Länge des Datenfeldes an. Ist es beim Öffnen des Device nicht Null, so wird versucht, für die in *ioa_Data* stehende Kanalmaske Speicherplatz bereitzustellen.

❑ **ioa_Period 0x002A (42)** Hier steht die zu spielende Frequenz. Dieses Feld wird beim *ACMD_PERVOL*-Befehl benutzt.

❑ **ioa_Volume 0x002C (44)** Gibt beim *ACMD_PERVOL*-Befehl die Lautstärke des Tones an.

❑ **ioa_Cycles 0x002E (46)** Gibt die Anzahl der zu spielenden Zyklen an. Wird bei der Tonausgabe mit CMD_WRITE benutzt.

❑ **ioa_WriteMessage 0x0030 (48)** Eine normale *Message*-Datenstruktur wie sie im Kapitel über »Exec« beschrieben ist. Dieses Feld hat nur im Zusammenhang mit dem *ADIOF_WRITEMESSAGE*-Flags eine Bedeutung und sagt Ihnen, wann die Bearbeitung des *CMD_WRITE*-Befehls aufgenommen wurde.

8.2 Das Clipboard-Device

Das Clipboard-Device dient zur einfachen Verwaltung und Zwischenspeicherung von Datenblöcken. Die Programmierung von Blockoperationen, wie sie aus der Textverarbeitung bekannt sind, sind damit ohne größere Umstände zu realisieren.

ClipboardUnitPartial

Offset		C (devices/clipboard.h)	Modula-2 (Clipboard)
Hex	Dez	struct ClipboardUnitPartial	ClipboardUnitPartial =
		{	RECORD
0000	0	struct Node cu_Node;	node:Node;
000E	14	ULONG cu_UnitNum;	unitNum:LONGCARD;
0012	18	};	END;

Datenstrukturenreferenz »ClipBoard«: *IOClipReq*.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur spezifiziert ein ClipBoard-Unit.

❑ **cu_Node 0x0000 (0)** Eine *Node*-Datenstruktur, die zur Verkettung mehrerer Units zu einer Liste dient.

❑ **cu_UnitNum 0x000E (14)** Die Nummer dieses Units.

8

IOClipReq

Offset		C (devices/clipboard.h)	Modula-2 (Clipboard)
Hex	Dez	struct IOClipReq	IOClipboard =
		{	RECORD
0000	0	struct Message io_Message;	message:Message;
0014	20	struct Device *io_Device;	device:DevicePtr;
0018	24	struct Unit *io_Unit;	unit:UnitPtr;
001C	28	UWORD io_Command;	command:CARDINAL;
001E	30	UBYTE io_Flags;	flags:IOFlagSet;
001F	31	BYTE io_Error;	error:Byte;
0020	32	ULONG io_Actual;	actual:LONGCARD;
0024	36	ULONG io_Length;	length:LONGCARD;
0028	40	STRPTR io_Data;	data:ADDRESS;
002C	44	ULONG io_Offset;	offset:LONGCARD;
0030	48	LONG io_ClipID;	clipID:LONGINT;
0034	52	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Dies ist eine erweiterte *IOStdReq*-Datenstruktur, die zur Steuerung und Übermittlung von Daten von und zum Clipboard-Device benutzt wird.

- **io_Message 0x0000 (0)** Eine *Message*-Datenstruktur, die von *DoIO* und *SendIO* benötigt wird.
- **io_Device 0x0014 (20)** Ein Adreßzeiger auf eine *DeviceNode*-Datenstruktur, durch die das Clipboard-Device in die Liste der Devices eingebunden wird.
- **io_Unit 0x0018 (24)** Ein Adreßzeiger auf die zu dieser Unit gehörenden *ClipboardUnitPartial*-Datenstruktur.
- **io_Command 0x001C (28)** Folgende Standardbefehle haben in diesem Device eine Funktion (zu den Definitionen in C und Assembler gehört jeweils das Präfix *CMD_*):

C	Modula-2	Dez	Beschreibung
RESET	reset	1	Neuinitialisierung des ClipBoard-Device.
READ	read	2	Einlesen eines Clips in den Datenpuffer.

C	Modula-2	Dez	Beschreibung
WRITE	write	3	Schreiben eines Clips aus dem Datenpuffer.
UPDATE	update	4	Wird nach Beendigung von Schreibzugriffen gesendet, um die Vollständigkeit des Clips zu signalisieren.

Weiterhin sind folgende Befehle zusätzlich definiert (zu den Definitionen in C und Assembler gehört jeweils das Präfix *CBD_*):

C	Modula-2	Dez	Beschreibung
POST	post	9	Einen Messageport, an den *SatisfyMsg*-Datenstrukturen geschickt werden können, anmelden. Dann können die Clipboard-Device Daten anfordern, falls sie gebraucht werden. Wenn ein Message ankommt, muß sofort *WRITE* ausgeführt werden. Dabei muß *io_ClipID* auf den in *SatisfyMsg* stehenden Wert gesetzt werden.
CLIPREADID	currentReadID	10	Das aktuelle ClipID für Lesen ins *io_ClipID* schreiben. Somit kann geprüft werden, ob die Clipdaten noch gültig sind.
CLIPWRITEID	currentWriteID	11	Das aktuelle ClipID für Schreiben *ins io_ClipID*. Ist diese größer als die in *SatisfyMsg*, dann braucht nicht mehr geschrieben zu werden, da die Daten veraltet sind.

8

- **io_Flags 0x001E (30)** Die Flags für die Bearbeitung der IO-Befehle (siehe »Exec«).
- **io_Error 0x001F (31)** Tritt ein Fehler auf, so steht hier ein von null verschiedener Wert.
- **io_Actual 0x0020 (32)** Die Anzahl der Bytes, die tatsächlich zum oder vom Datenpuffer transferiert wurden.
- **io_Length 0x0024 (36)** Die Anzahl der Bytes, die zum oder vom Datenpuffer transferiert werden sollen.
- **io_Data 0x0028 (40)** Ein Adreßzeiger auf die zu transferierenden Daten. Bei dem *POST*-Befehl steht hier ein Adreßzeiger auf eine *MsgPort*-

Datenstruktur, an die Anforderungen in Form von *SatisfyMsg*-Datenstrukturen geschickt werden können.

❑ **io_Offset 0x002C (44)** Die aktuelle Position innerhalb des Clips ???

❑ **io_ClipID 0x0030 (48)** Die Identifizierungsnummer des aktuellen *WRITE-* oder *READ*-Befehls. Findet der WRITE-Befehl als Antwort auf eine *SatisfyMsg*-Datenstruktur statt, so muß hier der gleiche Wert wie in dieser *SatisfyMsg*-Datenstruktur stehen.

SatisfyMsg

Offset		C (devices/clipboard.h)	Modula-2 (Clipboard)
Hex	Dez	struct SatisfyMsg	SatisfyMsg =
		{	RECORD
0000	0	struct Message sm_Msg;	msg:Message;
0014	20	UWORD sm_Unit;	unit:CARDINAL;
0016	22	LONG sm_ClipID;	clipID:LONGINT;
001A	26	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

8

Beschreibung

Diese Datenstruktur dient zur Anforderung von Daten durch das Clipboard-Device. Wenn eine »Post« mit dem *POST*-Befehl eingerichtet wurde, schickt das Clipboard-Device an den dort angegebenen Messageport eine Nachricht, sobald jemand Daten anfordert. Der Empfänger muß dann mit dem *WRITE*-Befehl sofort die benötigten Daten zu dem Clipboard-Device senden.

❑ **sm_Msg 0x0000 (0)** Eine normale *Message*-Datenstruktur, die zum Senden der *SatisfyMsg*-Datenstruktur mit Hilfe von *PutMsg* benötigt wird.

❑ **sm_Unit 0x0014 (20)** Gibt die Unit des Clipboard-Devices an, das die Daten anfordert.

❑ **sm_ClipID 0x0016 (22)** Die Identifizierung des Datenblocks, der zu der »Post« gehört.

8.3 Das Console-Device

Dieses Device ist für die Ein- und Ausgabe in dem Terminal-Fenster zuständig. Mit Hilfe von Sonderzeichensequenzen, die durch den Wert CSI (ASCII-Wert: 155) eingeleitet werden, ist nicht nur einfaches Einlesen und Schreiben möglich, sondern auch komplexe Anzeigengestaltung (Cursor-Steuerung,

Farben, Schriftart etc.). Beim Öffnen dieses Devices wird der Adreßzeiger auf die entsprechende Window-Datenstruktur im *io_Data*-Feld der an die Routine *OpenDevice* übergebenen IOStdReq-Datenstruktur eingetragen. Für dieses Device sind folgende Standard-Befehle zulässig:

C	Modula-2	Dez	Beschreibung
CMD_READ	read	2	Eine Zeichenkette in den Puffer einlesen.

io_Data: Ein Adreßzeiger auf den Puffer.
io_Length: Die Größe des Puffers in Bytes.
io_Actual Die Anzahl der tatsächlich gelesenen Zeichen.

C	Modula-2	Dez	Beschreibung
CMD_WRITE	write	3	Eine Zeichenkette aus den Puffer ausgeben. Das Zeichen CSI (155) leitet eine Steuersequenz ein.

io_Data: Ein Adreßzeiger auf den Puffer.
io_Length: Die Größe des Puffers in Bytes.

Folgende Befehle sind in diesem Device zusätzlich definiert:

C	Modula-2	Dez	Beschreibung
CD_ASKKEYMAP	askKeyMap	9	Einen Adreßzeiger auf die aktuelle Keymap (des Fensters) in *io_Data* holen.
CD_SETKEYMAP	setKeyMap	10	Eine neue Keymap für diese Fenster setzen.
CD_ASKDEFAULTKEYMAP	askDefaultKeyMap	11	Die Default- Keymap (aus den Preferences) holen.
CD_SETDEFAULTKEYMAP	setDefaultKeyMap	12	Eine neue Default-Keymap setzen.

8

Die CSI-Befehlssequenzen

Eine CSI-Befehlssequenz hat das folgende Format:

CSI-Parameter-Befehl
Beispiel: *CSI14;80H* (ASCII-Folge: *155,"14;80H"*)

CSI ist das Zeichen mit dem ASCII-Code 155 (Auf der Tastatur erzeugt durch die Tastenfolge [Esc] [[]).
Die Parameter werden als zeichenweise ausgeschriebene Zahlen übergeben. Mehrere Parameter werden durch ein Semikolon getrennt. Abschließend folgt das ASCII-Zeichen eines der folgenden Befehle:

Befehl	Parameter	Bedeutung
@	n	n-Leerzeichen einfügen.
A	n	Cursor n-Zeilen aufwärts positionieren.
B	n	Cursor n-Zeilen abwärts positionieren.
C	n	Cursor n-Zeilen rechts positionieren.
D	n	Cursor n-Zeilen links positionieren.
E		Einen Zeilenvorschub durchführen.
F		Gegenteil von E.
H	x,y	Cursor auf Position (x,y) setzen.
I	n	Cursor n-Tabulatorschritte nach rechts.
J		Löschen von Cursorposition bis Displayende.
K		Löschen von Cursorposition bis Zeilenende.
L		Eine Zeile einfügen.
M		Eine Zeile löschen.
P	n	n-Zeichen ab Cursorposition löschen.
S	n	n-Zeilen nach oben scrollen.
T	n	n-Zeilen nach unten scrollen.
W	n	n-Tabulator-Stops setzen.
Z	n	Cursor n-Tabulatorschritte nach links.
g		Tabulator-Einstellungen löschen.
h		LineFeed als LineFeed + Return behandeln.
l		LineFeed als solchen behandeln.
m	s,f,b	Schriftstil, Vorder- und Hintergrundfarbe setzen (siehe Konstanten).
n		Beim nächsten CMD_READ-Befehl IDCMP-Status lesen.
p	bool	bool = 0 Cursor aus-, bool = 1 Cursor einschalten.
t	n	Die Seitenlänge auf n-Zeilen setzen.
u	n	Die Zeilenlänge auf n-Zeichen setzen.
x	n	Den linken Offset auf n-Zeichen setzen.
y	n	Den oberen Offset auf n-Zeichen setzen.

In dem Include-File zur Console-Device sind jede Menge Konstanten definiert, die im Zusammenhang mit den CSI-Kommando »m« benutzt werden. Sie definieren Werte für die Schriftart, sowie Vorder- und Hintergrundfarbe.

C	Modula-2	Dez	Bedeutung
SGR_PRIMARY	primary	0	Normales Schriftbild
SGR_BOLD	bold	1	Fettschrift
SGR_ITALIC	italic	3	Schrägschrift
SGR_UNDERSCORE	underscore	4	Unterstrichen
SGR_NEGATIVE	negative	7	Weiß auf schwarz schreiben
SGR_BLACK	black	30	Vordergrundfarbe schwarz
SGR_RED	red	31	Vordergrundfarbe rot
SGR_GREEN	green	32	Vordergrundfarbe grün
SGR_YELLOW	yellow	33	Vordergrundfarbe gelb
SGR_BLUE	blue	34	Vordergrundfarbe blau

C	Modula-2	Dez	Bedeutung
SGR_MAGENTA	magenta	35	Vordergrundfarbe magentarot
SGR_CYAN	cyan	36	Vordergrundfarbe zyan
SGR_WHITE	white	37	Vordergrundfarbe weiß
SGR_DEFAULT	default	39	Default-Vordergrundfarbe
SGR_BLACKBG	blackBg	40	Hintergrundfarbe schwarz
SGR_REDBG	redBg	41	Hintergrundfarbe rot
SGR_GREENBG	greenBg	42	Hintergrundfarbe grün
SGR_YELLOWBG	yellowBg	43	Hintergrundfarbe gelb
SGR_BLUEBG	blueBg	44	Hintergrundfarbe blau
SGR_MAGENTABG	magentaBg	45	Hintergrundfarbe magentarot
SGR_CYANBG	cyanBg	46	Hintergrundfarbe zyan
SGR_WHITEBG	whiteBg	47	Hintergrundfarbe weiß
SGR_DEFAULTBG	defaultBg	49	Default-Hintergrundfarbe

Folgende Befehle sind durch ein Zeichen definiert und können direkt aufgerufen werden:

Wert	Name	Auswirkung im Fenster
7	BEL – Bell	Einen Piepston erzeugen.
8	BS – BackSpace	Zeichen links vom Cursor löschen.
9	HT – HorizontalTab	Einen Tabulatorschritt nach rechts.
10	LF – LineFeed	Einen Zeilenvorschub.
11	VT – VerticalTab	Eine Zeile nach oben.
12	FF – FormFeed	Das Fenster löschen.
13	CR – CarrigeReturn	Zurück zum Zeilenanfang.

8.4 Das Gameport-Device

Mit diesem Device ist es möglich, die Gameports abzufragen. An den Gameports können Eingabegeräte wie Maus, Joysticks, Trackballs etc. angeschlossen werden. Befehle und Fehlermeldungen werden wieder über eine *IOStdReq*-Datenstruktur verwaltet. Der Parameter Unit bestimmt dabei, um welchen der beiden Ports es sich handelt (0 oder 1). Es steht nur folgender Befehl in diesem Device standardmäßig zur Verfügung:

C	Modula-2	Dez	Beschreibung
CMD_CLEAR	clear	5	Löscht den Eingabepuffer.

Folgende zusätzliche Befehle stehen in diesem Device zur Verfügung:

C	Modula-2	Dez	Beschreibung
GPD_READEVENT	readEvent	9	Status des Eingabegeräts lesen.

io_Data: Der Aderßzeiger auf eine Kette von *InputEvent*-Datenstrukturen, aus deren Komponenten die Zustände des Eingabegeräts gelesen werden können.
io_Length: Die Anzahl der *InputEvent*-Datenstrukturen mal 22.

GPD_ASKCTYPE	askCType	10	Typ des Eingabegeräts lesen.

io_Data: Ein Adreßzeiger auf ein Byte, dessen Wert den Typ des Eingabegerätes festlegt.
io_Length: Also wohl mindestens 1.

GPD_SETCTYPE	setCType	11	Typ des Eingabegeräts setzen.

io_Data: Ein Adreßzeiger auf eine Byte, dessen Wert den Typ des Eingabegerätes festlegt.
io_Length: Also wohl mindestens 1.

GPD_ASKTRIGGER	askTrigger	12	Abfragen der Eingabebedingungen.

io_Data: Ein Adreßzeiger auf eine *GamePortTrigger*-Datenstruktur.
io_Length: Die Größe der *GamePortTrigger*-Datenstruktur (8).

GPD_SETTRIGGER	setTrigger	13	Setzen der Eingabebedingungen.

io_Data: Ein Adreßzeiger auf eine *GamePortTrigger*-Datenstruktur.
io_Length: Die Größe der *GamePortTrigger*-Datenstruktur (8).

Folgende Eingabegeräte können über io_Data gesetzt werden (zu den Definitionen in C und Assembler gehört jeweils das Präfix *GPCT_*; in Modula-2 gehören die Parameter zu dem Aufzählungstypen *Controller*):

C	Modula-2	Dez	Beschreibung
NOCONTROLLER	noController	0	Es ist kein Eingabegerät angeschlossen.

C	Modula-2	Dez	Beschreibung
MOUSE	mouse	1	Das Eingabegerät ist eine Maus.
RELJOYSTICK	relJoystick	2	Das Eingabegerät ist ein relativer Joystick.
ABSJOYSTICK	absJoystick	3	Das Eingabegerät ist ein normaler Joystick.

GamePortTrigger

Offset		C (devices/gameport.h)	Modula-2 (GamePort)
Hex	Dez	struct GamePortTrigger	GamePortTrigger =
		{	RECORD
0000	0	UWORD gpt_Keys;	keys:KeySet;
0002	2	UWORD gpt_Timeout;	timeout:CARDINAL;
0004	4	UWORD gpt_XDelta;	xDelta:CARDINAL;
0006	6	UWORD gpt_YDelta;	yDelta:CARDINAL;
0008	8	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

8

Beschreibung

Durch diese Datenstruktur werden einige Zusatzbedingungen des zugehörenden Gameports festgelegt.

❏ **gpt_Keys 0x0000 (0)** Über diesen Parameter wird festgelegt, ob das Drücken oder das Loslassen der Tasten des Eingabegeräts gemeldet wird. Dabei können auch beide der folgenden Bits gesetzt werden:

C	Modula-2	Dez	Beschreibung
GPTF_DOWNKEYS	downKeys	1	Das Drücken der Tasten wird gemeldet.
GPTF_UPKEYS	upKeys	2	Das Loslassen der Tasten wird gemeldet.

- **gpt_Timeout 0x0002 (2)** Dieser Wert gibt an, nach wie vielen Fünfzigstel Sekunden ein »ReadEvent« vom Gameport-Device abgeschickt wird.
- **gpt_XDelta 0x0004 (4)** Dieser Wert gibt die Anzahl von Impulsen in x-Richtung an, die als ein einziger interpretiert werden. Ist eine Maus an dem Gameport angeschlossen, so kann über diesen Wert die Genauigkeit der Abfrage angegeben werden. Je größer dieser Wert ist, desto weiter muß die Maus bewegt werden, um einen tatsächlichen Impuls auszulösen.
- **gpt_YDelta 0x0006 (6)** Dieser Wert gibt die Anzahl von Impulsen in y-Richtung an, die als ein einziger interpretiert werden (siehe auch *gpt_XDelta*).

8.5 Das Input-Device

Das Input-Device verfügt zwar nicht über eigene Datenstrukturen, jedoch werden wie bei den anderen Devices mehrere Befehle über eine *IOStdReq*-Datenstruktur bereitgestellt. Folgende Standardbefehle stehen in diesem Device zur Verfügung:

C	Modula-2	Dez	Beschreibung
CMD_RESET	reset	1	Ein Reset durchführen. Dabei werden keine Handler deaktiviert.
CMD_STOP	stop	6	Dieser Befehl stoppt das Input Device.
CMD_START	start	7	Input-Device nach *CMD_STOP* wieder »anschmeißen«.

Die folgenden Befehle sind für das Input-Device zusätzlich definiert (zu den Definitionen in C und Assembler gehört jeweils das Präfix *IND_*):

C	Modula-2	Dez	Beschreibung
ADDHANDLER	addHandler	9	Fügt einen neuen Input-Handler ein. So können Sie die Input-Events abfangen, bevor sie das restliche System erreichen. In welcher Reihe die Handler die Events bekommen, entscheidet die Priorität. Der Systemhandler hat die Priorität 50.

io_Data: Ein Adreßzeiger auf eine *Interrupt*-Datenstruktur, deren Felder wie folgt initialisiert werden müssen:

is_Code: Ein Adreßzeiger auf den Code des Handlers (der Anfang muß in Assembler geschrieben sein, da die Parameter nicht auf den Stack, sondern in den Registern übergeben werden).

is_Data: Ein Adreßzeiger auf den privaten Datenbereich des Handlers. Diese Adresse wird beim Aufruf des Handlers im Adreßregiser *A1* übergeben.

is_Node.ln_Pri: Die Priorität des Handlers.

C	Modula-2	Dez	Beschreibung
REMHANDLER	remHandler	10	Einen durch *IND_ADDHANDLER* gesetzten Input-Handler aus der Liste entfernen.

io_Data: Ein Adreßzeiger auf den zu entfernenden Handler (siehe *IND_ADDHANDLER*).

WRITEEVENT	writeEvent	11	Leitet alle Input-Events an alle Handler weiter.

io_Data: Ein Adreßzeiger auf die verketteten *InputEvent*-Datenstrukturen.

io_Length: Die Länge der *InputEvent*-Datenstruktur in Bytes, also 22.

SETTHRESH	setThresh	12	Setzt die Anlaufzeit der Tastenwiederholung neu.

8

io_tv_secs: Der Sekundenanteil der Zeitdauer, die eine Taste gedrückt sein muß, bevor sie periodisch wiederholt wird. Wollen Sie die Tastenwiederholung tatsächlich benutzen, so dürfte hier 0 der einzig richtige Eintrag sein.

io_tv_Nicro: Der Mikrosekundenanteil der Zeitdauer, die eine Taste gedrückt sein muß, bevor sie periodisch wiederholt wird.

SETPERIOD	setPeriod	13	Setzt die Tastenwiederholgeschwindigkeit neu.

io_tv_secs: Die Anzahl der Sekunden, die zwischen der periodischen Wiederholung einer Taste verstreichen dürfen (sinnvollerweise 0).

io_tv_Micro: Der Mikrosekundenanteil der Zeitdauer, die bei der periodischen Wiederholung einer Taste zwischen den einzelnen Wiederholungen verstreichen darf.

C	Modula-2	Dez	Beschreibung
SETMPORT	setMPort	14	Setzt den aktuellen Gameport.

io_Data: Ein Adreßzeiger auf ein Byte, das 0 oder 1 sein muß. Dementsprechend wird der linke oder rechte Gameport als aktueller gesetzt.
io_Length: Muß 1 sein.

SETMTYPE	setMType	15	Legt den Typ des aktuellen Eingabegerätes fest.

io_Data: Ein Adreßzeiger auf ein Byte, dessen Inhalt den Typ des angeschlossenen Eingabegerätes festlegt. Dabei gelten die gleichen Definitionen wie im GamePort-Device.
io_Length: Sollte wohl nichts anderes als 1 sein.

SETMTRIG	setMTrig	16	Setzt die Eingabebedingungen für den aktuellen Gameport.

io_Data: Ein Adreßzeiger auf die *GamePortTrigger*-Datenstruktur, die die Eingabebedingungen beschreibt.
io_Length: Die Länge der *GamePortTrigger*-Datenstruktur, also 8.

8

8.6 Das Keyboard-Device

Das Keyboard-Device verfügt zwar nicht über eigene Datenstrukturen, jedoch werden wie bei den anderen Devices mehrere Befehle über eine *IOStdReq*-Datenstruktur bereitgestellt. Diese werden wir hier kurz aufführen. Folgende Standardbefehle stehen in diesem Device zur Verfügung:

C	Modula-2	Dez	Beschreibung
CMD_RESET	reset	1	Setzt das KeyBoard-Device in den Ursprungszustand.
CMD_CLEAR	clear	5	Löschen des Tastaturpuffers.

Die folgenden Befehle sind zusätzlich definiert (zu den Definitionen in C und Assembler gehört jeweils das Präfix *KBD_*):

C	Modula-2	Dez	Beschreibung
READEVENT	readEvent	9	Lesen eines Tastatur-Events.

io_Data: Der Adreßzeiger auf eine Liste von *InputEvent*-Datenstrukturen.
io_Length: Die Anzahl der *InputEvent*-Datenstrukturen mal 22.

READMATRIX	readMatrix	10	Lesen der gesamten Tastaturmatrix.

io_Data: Ein Adreßzeiger auf einen Puffer, dessen einzelne Bits angeben, ob die korrespondierende Taste gedrückt ist (1) oder nicht (0).
io_Length: Muß groß genug sein, um den Puffer zu enthalten.

ADDRESETHANDLER	addResetHandler	11	Fügt eine Liste von Routinen vor die Systemreset-Routine ein.
REMRESETHANDLER	remResetHandler	12	Entfernt eine zuvor eingefügte Liste von Routinen.
RESETHANDLERDONE	resetHandlerDone	13	Signalisiert, daß die eigenen Reset-Routinen ausgeführt wurden.

BEACHTE: Der Amiga 500 und 2000 unterstützt nicht die eigenen Reset-Routinen. Durch <Ctrl> + <rAmiga> + <lAmiga> wird bei ihnen unmittelbar ein Hard-Reset durchgeführt.

8.7 Das Narrator-Device

Das Narrator-Device steuert die Sprachausgabe, also die akustische Ausgabe von Zeichenketten, die die dafür vorgesehenen Phoneme enthalten muß. Folgende Standardbefehle stehen in diesem Device zur Verfügung (zu den Definitionen in C und Assembler gehört jeweils das Präfix *CMD_*):

C	Modula-2	Dez	Beschreibung
RESET	reset	1	Unterbricht eine eventuell laufende Sprachausgabe und setzt alle Parameter auf die ursprünglichen Werte zurück.
READ	read	2	Liest die zu der augenblicklichen Aussprache gehörende Mundform.
WRITE	write	3	Eine Null-terminierte Phonem-Zeichenkette wird akustisch ausgegeben.
STOP	stop	6	Unterbricht die Sprachausgabe.

C	Modula-2	Dez	Beschreibung
START	start	7	Fährt nach der Unterbrechung durch

Für dieses Device sind darüber hinaus noch folgende Fehlermeldungen definiert (zu den Definitonen in C und Assembler gehört jeweils das Präfix *ND_*):

C	Modula-2	Dez	Beschreibung
NOMEM	moMem	-2	Es konnte kein Speicherplatz allokiert werden.
NOAUDLIB	noAudLib	-3	Das Audio-Device konnte nicht geöffnet werden.
MAKEBAD	makeBad	-4	Ein Fehler beim Aufruf der Make Library-Routine.
UNITERR	unitErr	-5	Eine andere Unitnummer als 0 ist unzulässig.
CANTALLOC	cantAlloc	-6	Die Audio-Kanäle konnten nicht allokiert werden.
UNIMP1	unimp1	-7	Nicht implementierter Befehl.
NOWRITE	noWrite	-8	Aufruf von *CMD_READ*, ohne daß vorher *CMD_WRITE* aufgerufen wurde.
EXPUNGED	expunged	-9	Das Device existiert nicht mehr.
PHONERR	phonErr	-20	Ein Fehler in der Phonem-Zeichen-kette.
RATEERR	rateErr	-21	Der Parameter rate liegt außerhalb des Gültigkeitsbereichs (40–400).
PITCHERR	pitchErr	-22	Der Parameter pitch liegt außerhalb des Gültigkeitsbereichs (65–320).
SEXERR	sexErr	-23	Der Parameter sex liegt außerhalb des Gültigkeitsbereichs (0 oder 1).
MODEERR	modeErr	-24	Der Parameter mode liegt außerhalb des Gültigkeitsbereichs (0 oder 1)
FREQERR	freqErr	-25	Der Parameter sampfreq liegt außerhalb des Gültigkeitsbereichs (5000–28000).
VOLERR	VolErr	-26	Der Parameter volume liegt außerhalb des Gültigkeitsbereichs (0–64).

narrator_rb

Offset		C (devices/narrator.h)	Modula-2 (Narrator)
Hex	Dez	struct narrator_rb	IONarrator =
		{	RECORD
0000	0	struct IOStdReq message;	message:IOStdReq;
0030	48	UWORD rate;	rate:CARDINAL;
0032	50	UWORD pitch;	pitch:CARDINAL;
0034	52	UWORD mode;	mode:CARDINAL;
0036	54	UWORD sex;	sex:CARDINAL;
0038	56	UBYTE *ch_masks;	chMasks:ADDRESS;
003C	60	UWORD nm_masks;	nmMasks:CARDINAL;
003E	62	UWORD volume;	volume:CARDINAL;
0040	64	UWORD sampfreq;	sampFreq:CARDINAL;
0042	66	UBYTE mouths;	mouths:UByte;
0043	67	UBYTE chanmask;	chanMask:UByte;
0044	68	UBYTE numchan;	numChan:UByte;
0045	69	UBYTE pad;	pad:BYTE;
0046	70	};	END;

Assembler Präfix: *NDI_*

Syntax: Strukturname = *NDI*

Datenstrukturenreferenz Keine.

Routinenreferenz »Exec«: *OpenDevice*.

8

Beschreibung

Wie bei den anderen Devices ist dies eine erweiterte *IOStdReq*-Datenstruktur, die noch zusätzliche Informationen über die Art der Stimme beinhaltet. Zur Steuerung der Stimme stehen wieder einige Flags und Befehle zur Verfügung. Interessant ist die Möglichkeit über *CMD_READ* Informationen über die aktuelle Mundform abzufragen (siehe auch die hierauf folgende Datenstruktur *mouth_rb*).

- **message 0x0000 (0)** Eine eingebundene *IOStdReq*-Datenstruktur, über die die Verbindung zu dem Device hergestellt wird.
- **rate 0x0030 (48)** Dieser Wert bestimmt die Sprechgeschwindigkeit. Er entspricht in etwa den gesprochenen Wörtern pro Minute, gültige Werte liegen zwischen 40 und 400.
- **pitch 0x0032 (50)** Die mittlere Tonlage der Sprachausgabe, gültige Werte liegen zwischen 65 und 320.
- **mode 0x0043 (52)** Folgende zwei Konstanten definieren die Modi für die Art der Betonung:

C	Modula-2	Dez	Beschreibung
NATURALF0	natural	0	Ergibt eine menschliche Sprechweise mit entsprechenden Betonungen.
ROBOTICF0	robotic	1	Ergibt eine monotone, gleichmäßige Betonung.

❑ **sex 0x0036 (54)** Folgende zwei Konstanten definieren das Geschlecht des Sprechers:

C	Modula-2	Dez	Beschreibung
MALE	male	0	Es wird eine männliche Stimme simuliert.
FEMALE	female	1	Es wird eine weibliche Stimme simuliert.

❑ **ch_masks 0x0038 (56)** Ein Adreßzeiger auf eine Tabelle mit Kanalmasken, die definieren, welche Ausgabekanäle für die Sprachausgabe benutzt werden (siehe auch »Audio«).

❑ **nm_masks 0x003C (60)** Die Anzahl der Kanalmasken, auf die *ch_masks* zeigt (siehe auch »Audio«).

❑ **volume 0x003E (62)** Über diesen Wert wird die durchschnittliche Lautstärke der Sprachausgabe festgelegt, gültige Werte liegen zwischen 0 und 64.

❑ **sampfreq 0x0040 (64)** Die Sampling-Frequenz, mit der die Sprachausgabe erfolgt, gültige Werte liegen zwischen 5000 und 28000.

❑ **mouths 0x0042 (66)** Steht an dieser Stelle 1, so wird eine Abfrage der zu einem Phonem gehörenden Mundform ermöglicht (siehe die hierauf folgende *mouth_rb*-Datenstruktur).

❑ **chanmask 0x0043 (67)** Ein systeminterner Adreßzeiger für Audio-Allocation-Masks.

❑ **numchan 0x0044 (68)** Ein systeminterner Wert für die Anzahl der in *chanmask* benutzten Masken.

❑ **pad 0x0045 (69)** Ein Dummerchen... (unbedeutender Füllwert)

mouth_rb

Offset		C (devices/narrator.h)	Modula-2 (Narrator)
Hex	Dez	struct mouth_rb	Mouth =
		{	RECORD
0000	0	struct narrator_rb voice;	voice:IONarrator;
0046	70	UBYTE width;	width:UByte;
0047	71	UBYTE height;	height:UByte;
0048	72	UBYTE shape;	shape:BYTE;
0049	73	UBYTE pad;	pad:BYTE;
0050	74	};	END;

Assembler Präfix: *MRB_*
Syntax: Strukturname = *MRB*

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Durch diese Datenstruktur werden zu einer Stimme Daten über die augenblickliche Mundform bereitgestellt. Damit ist es möglich, grafisch einen Mund zu simulieren, dessen Bewegungen mit der Stimme zusammenpassen.

- **voice 0x0000 (0)** Eine eingebundene *IONarrator*-Datenstruktur, die die Stimme definiert, zu der Informationen über die Mundform erwünscht sind.
- **width 0x0046 (70)** Dieser Wert entspricht der momentanen Breite des Mundes.
- **height 0x0047 (71)** Dieser Wert entspricht der momentanen Höhe des Mundes.
- **shape 0x0048 (72)** Ein Systembyte, von dem Sie die Finger lassen sollten...
- **pad 0x0049 (73)** Ein Dummerchen... (unbedeutender Füllwert)

8

8.8 Das Parallel-Device

Das Parallel-Device übernimmt die Steuerung der parallelen Schnittstelle am Amiga. Die Kommunikation mit diesem Device läuft wieder über eine *IOStdReq*-Datenstruktur. In *io_Data* muß der Adreßzeiger auf dem Speicherbereich stehen, aus dem bzw. in den gelesen werden soll, und in *io_Length* die Länge dieses Speicherbereiches in Bytes. Folgende Standardbefehle stehen zur Verfügung (zu den Definitionen in C und Assembler gehört das Präfix *CMD_*):

C	Modula-2	Dez	Beschreibung
RESET	reset	1	Führt einen Reset der parallelen Schnittstelle durch.
READ	read	2	Liest Daten von der parallelen Schnittstelle.
WRITE	write	3	Sendet Daten zur parallelen Schnittstelle.
STOP	stop	6	Unterbricht die Datenübertragung.
START	start	7	Führt die durch *STOP* unterbrochene Datenübertragung wieder fort.
FLUSH	flush	8	Löscht alle zur Ein- und Ausgabe bereitstehenden »I/O-Requests«.

Für das Parallel-Device sind folgende Befehle zusätzlich definiert (zu den Definitionen in C und Assembler gehört das Präfix *PDCMD_*):

C	Modula-2	Dez	Beschreibung
QUERRY	querry	9	Ermittelt den aktuellen Port-Status (*io_Status*).
SETPARAMS	setParams	10	Setzt neue Parameter für die Schnittstelle.

Folgende Fehlermeldungen sind definiert (zu den Definitionen in C und Assembler gehört das Präfix *ParErr_*):

C	Modula-2	Dez	Beschreibung
DEVBUSY	devBusy	1	Das Parallel-Device arbeitet gerade.
BUFTOBIG	buffToBig	2	Der Zwischenspeicher ist zu groß.
INVPARAM	invParam	3	Die gewählte Parameterveränderung ist nicht erlaubt.
LINEERR	lineErr	4	Ein Übertragungsfehler ist aufgetreten.
NOTOPEN	notOpen	5	Das Parallel-Device konnte nicht geöffnet werden.
PORTRESET	portReset	6	An der parallelen Schnittstelle wird ein Reset durchgeführt.
INITERR	initErr	7	Die Initialisierung des Parallel-Devices verlief nicht erfolgreich.

IOExtPar

Offset		C (devices/parallel.h)	Modula-2 (Parallel)
Hex	Dez	struct IOExtPar	IOExtPar =
		{	RECORD
0000	0	struct IOStdReq IOPar;	ioPar:IOStdReq;
0030	48	ULONG io_PExtFlags;	pExtFlags:LONGCARD;
0034	52	UBYTE io_Status;	status:StatusSet;
0035	53	UBYTE io_ParFlags;	parFlags:ParFlagSet;
0036	54	struct IOPArray io_PTermArray;	pTermArray:IOPArray;
003E	62	};	END;

Assembler — Syntax: Strukturname = *IOEXTPAR*

Definition der einzelnen Parameter in Versalien.

Datenstrukturenreferenz — Keine.

Routinenreferenz — »Exec«: *OpenDevice.*

Beschreibung

Bei der *IOExtPar*-Datenstruktur handelt es sich um eine für das Parallel-Device erweiterte *IOStdReq*-Datenstruktur. Für den Datenaustausch stehen insbesondere die Standard-Befehle der *IOStdReq*-Datenstruktur zur Verfügung.

8

- **IOPar 0x0000 (0)** — Eine eingebundene *IOStdReq*-Datenstruktur. Die zur Verfügung stehenden Befehle und Fehlermeldungen sind am Anfang dieses Abschnitts beschrieben.

- **io_PExtFlags 0x0030 (48)** — Eine in den vorliegenden Versionen unbenutzte 4-Byte-Marke.

- **io_Status 0x0034 (52)** — In diesem Feld steht nach der Anwendung des Befehls *PDCMD_QUERY* der aktuelle Status der parallelen Schnittstelle bzw. des angeschlossenen Geräts. In älteren Versionen sind *IOPTB_PARBUSY* und *IOPTB_PARSEL* miteinander vertauscht. Zu den Definitionen in C gehört das Präfix: *IOPTB_*; in Assembler werden die Bits direkt definiert: *BITDEF IOPT*,<Name>,<Bit>;

C	Modula-2	Bit	Beschreibung
PARBUSY	pBusy	0	Bit gesetzt: Das Gerät tut gerade nichts. Bit gelöscht: Das Gerät arbeitet gerade.
PAPEROUT	PaperOut	1	Bit gesetzt: Soweit alles OK. Bit gelöscht: Drucker hat kein Papier.

C	Modula-2	Bit	Beschreibung
PARSEL	pSel	2	Bit gesetzt: Das Gerät ist im OFF-LINE-Zustand. Bit gelöscht: Das Gerät ist im ONLINE-Zustand.
RWDIR	rwDir	3	Bit gesetzt: Die Schnittstelle dient zur Ausgabe. Bit gelöscht: Die Schnittstelle dient zur Eingabe.

❑ **io_ParFlags 0x0035 (53)**

Folgende Bits haben in dieser 1-Byte-Marke gesetzt eine Bedeutung (zu den Definitionen in C gehört das Präfix: *PARB_*; in Assembler werden die Bits direkt definiert: *BITDEF PAR*,<Name>,<Bit>;):

C	Modula-2	Dez	Beschreibung
EOFMODE	eofMode	2	Als Terminator gilt nicht das Standard-Nullbyte, sondern es gelten die in *io_PTermrray* definierten Terminatoren.
SHARED	shared	32	Es können mehrere Tasks gleichzeitig auf die Schnittstelle zugreifen. Dieser Modus ist jedoch nicht zu empfehlen, da die Daten mehrerer Tasks bei der Ausgabe vermischt werden.

8

❑ **io_PTermArray 0x0036 (54)**

Eine eingebundene *IOPArray*-Datenstruktur, in der bis zu acht Terminatoren definiert werden können.

IOPArray

Offset Hex	Offset Dez	C (devices/parallel.h)	Modula-2 (Parallel)
		struct IOPArray	IOPArray =
		{	RECORD
0000	0	ULONG PTermArray0;	pTermArray0:LONGCARD;
0004	4	ULONG PTermArray1;	pTermArray1:LONGCARD;
0008	8	};	END;

Assembler

Syntax: Strukturname = *PTERMARRAY*
PtermArray0 —> PTERMARRAY_0
PtermArray1 —> PTERMARRAY_1

Datenstrukturenreferenz »Parallel«: *IOExtPar.*

Routinenreferenz Keine.

Beschreibung

In dieser Datenstruktur können bis zu acht Terminatoren definiert werden, bei deren Auftreten sofort der Schreib-Lese-Vorgang abgebrochen wird. Werden weniger als acht Terminatoren definiert, so müssen die freien Stellen mit dem Wert des letzten Terminators belegt werden.

- **PTermArray0 0x0000 (0)** In diesem 4 Byte langen Wert werden die ersten vier Terminatoren definiert.
- **PTermArray1 0x0004 (4)** In diesem 4 Byte langen Wert werden die letzten vier Terminatoren definiert.

8.9 Das Printer-Device

Wie nicht anders zu erwarten, läuft über dieses Device die eigentliche Druckerausgabe. Dazu gehört insbesondere das Senden von Druckerkommandos sowie die Text- und Grafikausgabe auf dem Drucker. Für druckerspezifische Anpassungen ist dieses Device jedoch nicht zuständig, dazu gibt es das PrtBase-Device. Für die einfache Textausgabe reicht eine normale *IOStdReq*-Datenstruktur, in der die Kommandos *CMD_WRITE* (3) und *PRD_RAWWRITE* (9) verwendet werden können. Folgende Standardbefehle sind in diesem Device zulässig:

C	Modula-2	Dez	Beschreibung
CMD_INVALID	invalid	0	Setzt das Device in den Error-Zustand.
CMD_RESET	reset	1	Führt am Drucker ein Reset durch.
CMD_WRITE	write	3	Text auf dem Drucker ausgeben.

io_Data: Ein Adreßzeiger auf die Zeichenkette, die auf den Drucker ausgegeben wird.
io_Length : Die Länge der Zeichenkette oder –1, wenn es eine Null-terminierte ist.

C	Modula-2	Dez	Beschreibung
CMD_STOP	stop	6	Unterbricht die Ausgabe auf den Drucker.
CMD_START	start	7	Führt die unterbrochene Ausgabe wieder fort.
CMD_FLUSH	flush	8	Löschen aller bereitstehenden »IORequests«.

IOPrtCmdReq

Offset Hex	Dez	C (devices/printer.h)	Modula-2 (Printer)
		struct IOPrtCmdReq	IOPrtCmdReq =
		{	RECORD
0000	0	struct Message io_Message;	message:Message;
0014	20	struct Device *io_Device;	device:DevicePtr;
0018	24	struct Unit *io_Unit;	unit:UnitPtr;
001C	28	UWORD io_Command;	command:CARDINAL;
001E	30	UBYTE io_Flags;	flags:IOFlagSet;
001F	31	BYTE io_Error;	error:Error;
0020	32	UWORD io_PrtCommand;	prtCommand:CARDINAL;
0022	34	UBYTE io_Parm0;	parm0:UByte;
0023	35	UBYTE io_Parm1;	parm1:UByte;
0024	36	UBYTE io_Parm2;	parm2:UByte;
0025	37	UBYTE io_Parm3;	parm3:UByte;
0026	38	};	END;

Assembler Die ersten sechs Parameter (*IORequest*) sind nicht definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz »ExecSupport«: *CreateExtIO*, *DeleteExtIO*.

8

Beschreibung

Diese Datenstruktur ist eine für das Printer-Device abgewandelte IORequest-Datenstruktur. Mit ihr können dem Drucker Druckerkommandos zur Ausführung übergeben werden.

- **io_Message 0x0000 (0)** Eine eingebundene *Message*-Datenstruktur.
- **io_Device 0x0014 (20)** Ein Adreßzeiger auf eine *Device*-Datenstruktur, die dieses Device spezifiziert.
- **io_Unit 0x0018 (24)** Ein Adreßzeiger auf eine *Unit*-Datenstruktur.
- **io_Command 0x001C (28)** Es ist nur folgender Befehl definiert:

C	Modula-2	Dez	Beschreibung
PRD_PRTCOMMAND	prtCommand	10	Senden einer Kontrollsequenz.

- **io_Flags 0x001E (30)** (wie in *IOStdReq*-Datenstruktur)
- **io_Error 0x001F (31)** Folgende Fehlermeldungen sind extra definiert (zu den Definitionen in C und Assembler gehört jeweils das Präfix *PDERR_*):

C	Modula-2	Dez	Beschreibung
CANCEL	cancel	1	Durch Benutzer abgebrochen.
INTERNALMEMORY	internalMemory	6	Es ist nicht genug Speicherplatz für interne Variablen vorhanden.
BUFFERMEMORY	bufferMemory	7	Es ist nicht genug Speicherplatz für den Druckerpuffer vorhanden.

❑ **io_PrtCommand 0x0020 (32)** Folgende Kommandosequenzen (No.) sind definiert:

No.	C	Modula-2	Sequenz	Beschreibung
0	aRIS	ris	ESCc	Druckerreset
1	aRIND	rin	ESC#1	Druckerinitialisierung
2	aIND	ind	ESCD	lf (linefeed); Zeilenvorschub
3	aNEL	nel	ESCE	Return und Zeilenvorschub
4	aRI	ri	ESCM	Zeilenvorschub rückwärts
5	aSGR0	sgr0	ESC[0m	Schriftstil: normal
6	aSGR3	sgr3	ESC[3m	Schriftstil: kursiv (italic) ein
7	aSGR23	sgr23	ESC[23m	Schriftstil: kursiv (italic) aus
8	aSGR4	sgr4	ESC[4m	Schriftstil: unterstrichen ein
9	aSGR24	sgr24	ESC[24m	Schriftstil: unterstrichen aus
10	aSGR1	sgr1	ESC[1m	Schriftstil: fett ein
11	aSGR22	sgr22	ESC[22m	Schriftstil: fett aus
12	aSFC	sfc	SGR30-39	bei Farbdrucker: setzt Vordergrundfarbe
13	aSBC	sbc	SGR40-49	bei Farbdrucker: setzt Hintergrundfarbe
14	aSHORP0	shorp0	ESC[0w	Schriftart: Pica (normal)
15	aSHORP2	shorp2	ESC[2w	Schriftart: Elite ein
16	aSHORP1	shorp1	ESC[1w	Schriftart: Elite aus
17	aSHORP4	shorp4	ESC[4w	Schriftart: Schmalschrift ein
18	aSHORP3	shorp3	ESC[3w	Schriftart: Schmalschrift aus
19	aSHORP6	shorp6	ESC[6w	Schriftart: Breitschrift ein
20	aSHORP5	shorp5	ESC[5w	Schriftart: Breitschrift aus
21	aDEN6	den6	ESC[6"z	Schattenschrift ein
22	aDEN5	den5	ESC[5"z	Schattenschrift aus
23	aDEN4	den4	ESC[4"z	doppelter Anschlag ein
24	aDEN3	den3	ESC[3"z	doppelter Anschlag aus
25	aDEN2	den2	ESC[2"z	NLQ (NearLetterQuality) ein
26	aDEN1	den1	ESC[1"z	NLQ (NearLetterQuality) aus
27	aSUS2	sus2	ESC[2v	Hochgestellt ein
28	aSUS1	sus1	ESC[1v	Hochgestellt aus
29	aSUS4	sus4	ESC[4v	Tiefgestellt ein
30	aSUS3	sus3	ESC[3v	Tiefgestellt aus
31	aSUS0	sus0	ESC[0v	Teststellung normal
32	aPLU	plu	ESCL	Teillinie aufwärts

8

No.	C	Modula-2	Sequenz	Beschreibung
33	aPLD	pld	ESCK	Teillinie abwärts
34	aFNT0	fnt0	ESC(B	Zeichensatz: USA
35	aFNT1	fnt1	ESC(R	Zeichensatz: Frankreich
36	aFNT2	fnt2	ESC(K	Zeichensatz: Deutschland
37	aFNT3	fnt3	ESC(A	Zeichensatz: England
38	aFNT4	fnt4	ESC(E	Zeichensatz: Dänemark I
39	aFNT5	fnt5	ESC(H	Zeichensatz: Schweden
40	aFNT6	fnt6	ESC(Y	Zeichensatz: Italien
41	aFNT7	fnt7	ESC(Z	Zeichensatz: Spanien
42	aFNT8	fnt8	ESC(J	Zeichensatz: Japan
43	aFNT9	fnt9	ESC(6	Zeichensatz: Norwegen
44	aFNT10	fnt10	ESC(7	Zeichensatz: Dänemark II
45	aPROP2	prop2	ESC[2p	Proportionalschrift ein
46	aPROP1	prop1	ESC[1p	Proportionalschrift aus
47	aPROP0	prop0	ESC[0p	Proportionalschrift löschen
48	aTSS	tss	ESC[n E	Zeichenabstand n für Proportionalschrift
49	aJFY5	jfy5	ESC[5 F	Druckbild: linksbündig
50	aJFY7	jfy7	ESC[7 F	Druckbild: rechtsbündig
51	aJFY6	jfy6	ESC[6 F	Druckbild: Blocksatz ein
52	aJFY0	jfy0	ESC[0 F	Druckbild: Blocksatz aus
53	aJFY3	jfy3	ESC[3 F	Platzhalter für Proportionalschrift
54	aJFY1	jfy1	ESC[1 F	Zentrierung für Proportionalschrift
55	aVERP0	verp0	ESC[0z	Zeilenabstand 1/8"
56	aVERP1	verp1	ESC[1z	Zeilenabstand 1/6"
57	aSLPP	slpp	ESC[nt	Seitenlänge: n Zeilen
58	aPERF	perf	ESC[nq	Am Seitenende n Zeilen überspringen
59	aPERF0	perf0	ESC[0q	Am Seitenende keine Zeilen überspringen
60	aLMS	lms	ESC#9	linken Rand setzen
61	aRMS	rms	ESC#0	rechten Rand setzen
62	aTMS	tms	ESC#8	oberen Rand setzen
63	aBMS	bms	ESC#2	unteren Rand setzen
64	aSTBM	stbm	ESC[Pn1;	oberen (n1) und unteren (n2) Rand setzen Pn2r
65	aSLRM	slrm	ESC[Pn1;	linken (n1) und rechten (n2) Rand setzen Pn2r
66	aCAM	cam	ESC#3	Ränder löschen
67	aHTS	hts	ESCH	horizontalen Tabulator setzen
68	aVTS	vts	ESCJ	vertikalen Tabulator setzen
69	aTBC0	tbc0	ESC[0g	horizontalen Tabulator löschen
70	aTBC3	tbc3	ESC[3g	alle horizontalen Tabulatoren löschen
71	aTBC1	tbc1	ESC[1g	vertikalen Tabulator löschen
72	aTBC4	tbc4	ESC[4g	alle vertikalen Tabulatoren löschen
73	aTBCALL	tbcall	ESC#4	alle Tabulatoren löschen
74	aTBSALL	tbsall	ESC#5	Standard-Tabulatoren setzen
75	aEXTEND	extend	ESC[Pn"x	folgendes Zeichen ist ein Druckerkommando
76	aRAW		ESC[Pn"r	folgende Zeichen im RAW-Code

8

- **io_Parm0 0x0022 (34)** Erstes Zeichen der Steuersequenz.
- **io_Parm1 0x0023 (35)** Zweites Zeichen der Steuersequenz.
- **io_Parm2 0x0024 (36)** Drittes Zeichen der Steuersequenz.
- **io_Parm3 0x0025 (37)** Viertes Zeichen der Steuersequenz.

IODRPReq

Offset		C (devices/printer.h)	Modula-2 (Printer)
Hex	Dez	struct IODRPReq	IODRPReq =
		{	RECORD
0000	0	struct Message io_Message;	message:Message;
0014	20	struct Device *io_Device;	device:DevicePtr;
0018	24	struct Unit *io_Unit;	unit:UnitPtr;
001C	28	UWORD io_Command;	command:CARDINAL;
001E	30	UBYTE io_Flags;	flags:IOFlagSet;
001F	31	BYTE io_Error;	error:Error;
0020	32	struct RastPort *io_RastPort;	rastPort:RastPortPtr;
0024	36	struct ColorMap *io_ColorMap;	colorMap:ColorMapPtr;
0028	40	ULONG io_Modes;	modesHi:CARDINAL; (* : = Ø *)
			modes:ViewModeSet;
002C	44	UWORD io_SrcX;	srcX:CARDINAL;
002E	46	UWORD io_SrcY;	srcY:CARDINAL;
0030	48	UWORD io_SrcWidth;	srcWidth:CARDINAL;
0032	50	UWORD io_SrcHeight;	srcHeight:CARDINAL;
0034	52	LONG io_DestCols;	destCols:LONGINT;
0038	56	LONG io_DestRows;	destRows:LONGINT;
003C	60	UWORD io_Special;	special:SpecialSet;
003E	62	};	END;

8

Assembler Die ersten sechs Parameter (*IORequest*) sind nicht definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz »ExecSupport«: *CreateExtIO*, *DeleteExtIO*.

Beschreibung

Diese Datenstruktur ist eine für das Printer-Device abgewandelte *IORequest*-Datenstruktur. Mit ihr können Hardcopies von frei wählbaren Bildschirmausschnitten erstellt werden.

- **io_Message 0x0000 (0)** Eine eingebundene *Message*-Datenstruktur.
- **io_Device 0x0014 (20)** Ein Adreßzeiger auf eine *Device*-Datenstruktur, die dieses Device spezifiziert.
- **io_Unit 0x0018 (24)** Ein Adreßzeiger auf eine *Unit*-Datenstruktur.

❑ **io_Command 0x001C (28)** Der einzige zusätzlich definierte Befehl ist:

C	Modula-2	Dez	Beschreibung
PRD_DUMPRPORT	dumpRPort	11	Senden einer Kontrollsequenz.

❑ **io_Flags 0x001E (30)** (siehe *IOStdReq*-Datenstruktur).

❑ **io_Error 0x001F (31)** Folgende Fehlermeldungen sind extra definiert (zu den Definitionen in C und Assembler gehört jeweils das Präfix *PDERR_*):

C	Modula-2	Dez	Beschreibung
CANCEL	cancel	1	Durch Benutzer abgebrochen.
NOTGRAPHICS	notGraphics	2	Der angesprochene Drucker ist nicht grafikfähig.
INVERTHAM	invertHam	3	Im HAM-Modus ist kein Invertieren des Ausdrucks möglich.
BADDIMENSION	badDimension	4	Die angegebenen Dimensionen sind unzulässig.
DIMENSIONOVERFLOW	dimensionOverflow	5	Die angegebenen Dimensionen sind zu groß.
INTERNALMEMORY	internalMemory	6	Es ist nicht genug Speicherplatz für interne Variablen vorhanden.
BUFFERMEMORY	bufferMemory	7	Es ist nicht genug Speicherplatz für den Druckerpuffer vorhanden.

8

❑ **io_RastPort 0x0020 (32)** Ein Adreßzeiger auf die zu dem auszudruckenden Bildschirmausschnitt gehörende *RastPort*-Datenstruktur.

❑ **io_ColorMap 0x0024 (36)** Ein Adreßzeiger auf die zu dem auszudruckenden Bildschirmausschnitt gehörende *ColorMap*-Datenstruktur.

❑ **io_Modes 0x0028 (40)** In diesem Feld steht, in welchem View-Mode sich der ausgewählte Bildschirmausschnitt befindet.

❑ **io_SrcX 0x002C (44)** Die x-Koordinate der linken oberen Ecke des zu druckenden Bildschirmausschnitts.

❑ **io_SrcY 0x002E (46)** Die y-Koordinate der linken oberen Ecke des zu druckenden Bildschirmausschnitts.

❑ **io_SrcWidth 0x0030 (48)** Die Breite des zu druckenden Bildschirmausschnitts.

❑ **io_SrcHeight 0x0032 (50)**

Die Höhe des zu druckenden Bildschirmausschnitts.

❑ **io_DestCols 0x0034 (52)**

Dieser Wert bestimmt die Breite des Ausdrucks. Die Maßeinheit (einzelne Punkte oder 1/1000stel Zoll) wird durch die Flags in *io_special* definiert. Ist *io_DestCols* größer als 0, so bestimmt dieser Wert direkt die Breite des Ausdrucks, ist er gleich 0, so wird der Ausdruck die gesamte Papierbreite einnehmen. Interessant wird es für Werte kleiner als 0. In diesem Fall bestimmt der Betrag dieses Wertes im Zusammenhang mit *io_DestRows* einen Vergrößerungs-, bzw. Verkleinerungsfaktor für den Ausdruck. Der Faktor errechnet sich folgendermaßen:

$$\text{Faktor} := \frac{|io_DestCols|}{io_DestRows}$$

❑ **io_DestRows 0x0038 (56)**

Dieser Wert bestimmt die Höhe des Ausdrucks. Die Maßeinheit (einzelne Punkte oder 1/1000stel Zoll) wird durch die Flags in *io_special* definiert. Ist *io_DestRows* größer als 0, so bestimmt dieser Wert direkt die Höhe des Ausdrucks (solange *io_DestCols* größer oder gleich 0), ist er gleich 0, so wird der Ausdruck die gesamte Papierhöhe einnehmen.

❑ **io_Special 0x003C (60)**

Gesetzte Bits haben jeweils die folgende Bedeutung:

C	Modula-2	Hex	Beschreibung
MILCOLS	milCols	0001	Die Maßeinheit für *io_DestCols* ist 1/1000stel Zoll.
MILROWS	milRows	0002	Die Maßeinheit für *io_DestRows* ist 1/1000stel Zoll.
FULLCOLS	fullCols	0004	Der Ausdruck erhält die maximal mögliche Breite.
FULLROWS	fullRows	0008	Der Ausdruck erhält die maximal mögliche Höhe.
FRACCOLS	fracCols	0010	Breite des Ausdrucks = maximale Breite / *io_DestCols*.
FRACROWS	fracRows	0020	Höhe des Ausdrucks = maximale Höhe / *io_DestRows*.
CENTER	center	0040	Der Ausdruck wird zentriert.
ASPECT	aspect	0080	Der Ausdruck wird um 90 Grad gedreht.
DENSITY1	density1	0100	*DENSITY1* bis *DENSITY7*
DENSITY2	density2	0200	definieren die Zeichendichte. des
DENSITY4	density4	0400	Grafikausdrucks. Dabei entspricht die Endziffer. 1 der minimalen, die 7 der maximalen Zeichendichte.

8

C	Modula-2	Hex	Beschreibung
NOFORMFEED		0800	Am Ende des Ausdrucks wird kein Seitenvorschub (ff) ausgeführt.
TRUSTME		1000	Es wird kein Druckerreset ausgeführt.
NOPRINT		2000	Es werden zwar die Dimensions-Umrechnungen durchgeführt, jedoch wird kein Ausdruck ausgegeben. Somit können berechnete Werte nochmals überprüft und gegebenenfalls verändert werden.

Weiterhin sind noch folgende Konstanten zur Bestimmung der Zeichendichte definiert:

C	Modula-2	Hex	Beschreibung
DENSITY3 DENSITY5 DENSITY6 DENSITY7	density3 density5 density6 density7	0300 0500 0600 0700	DENSITY1 bis DENSITY7 definieren die Zeichendichte des Grafikausdrucks. Dabei entspricht die Endziffer 1 der minimalen, die 7 der maximalen Zeichendichte.

8.10 Das Serial-Device

Das Serial-Device übernimmt die Steuerung der seriellen Schnittstelle am Amiga. Die Kommunikation mit diesem Device läuft wieder über eine *IOStdReq*-Datenstruktur. In *io_Data* muß der Adreßzeiger auf den Speicherbereich stehen aus/in den gelesen werden soll und in *io_Length* die Länge dieses Speicherbereiches in Bytes. Folgende Standardbefehle stehen zur Verfügung (zu den Definitionen in C und Assembler gehört das Präfix *CMD_*):

C	Modula-2	Dez	Beschreibung
RESET	reset	1	Führt an der Schnittstelle ein Reset durch.
READ	read	2	Liest Daten von der seriellen Schnittstelle.

C	Modula-2	Dez	Beschreibung
WRITE	write	3	Sendet Daten zu der seriellen Schnittstelle.
CLEAR	clear	5	Setzt die Adreßzeiger der Lesepuffer zurück.
STOP	stop	6	Unterbricht die Datenübertragung (sendet ein xOFF-Signal an das angeschlossene Gerät).
START	start	7	Führt mit der durch *STOP* unterbrochenen Datenübertragung fort.
FLUSH	flush	8	Löscht die zur Ein- oder Ausgabe bereitstehenden »I/O-Requests«.

Für das Serial-Device sind folgende Befehle zusätzlich definiert (zu den Definitionen in C und Assembler gehört das Präfix: *SDCMD_*):

C	Modula-2	Dez	Beschreibung
QUERRY	query	9	Ermittelt den Status der Schnittstelle.
BREAK	break	10	Unterbricht die Datenübertragung.
SETPARAMS	setParams	11	Setzt für die Schnittstelle neue Parameter.

Für das Serial-Device sind folgende Fehlermeldungen definiert (zu den Definitionen in C und Assembler gehört das Präfix: *SerErr_*):

C	Modula-2	Dez	Beschreibung
DEVBUSY	devBusy	1	Schnittstelle überträgt gerade Daten.
BAUDMISMATCH	baudMismatch	2	Unbenutzt.
INVBAUD	invBaud	3	Unbenutzt.
BUFFERR	buffErr	4	Es ist nicht genug Speicherplatz für den Zwischenspeicher vorhanden.
INVPARAM	invParam	5	Die Parameter enthalten unzulässige Werte.
LINEERR	lineErr	6	Ein Übertragungsfehler ist aufgetreten.
NOTOPEN	notOpen	7	Unbenutzt.
PORTRESET	portReset	8	Unbenutzt.
PARITYERR	parityErr	9	Ein Paritätsfehler ist aufgetreten (die Prüfsumme stimmt nicht).
INITERR	initErr	10	Unbenutzt.
TIMEERR	timeErr	11	Ein Fehler in *io_BrkTime*.

C	Modula-2	Dez	Beschreibung
BUFOVERFLOW	bufOverflow	12	Der Zwischenspeicher ist zu klein.
NODSR	nodsr	13	Es gab kein DSR-Signal (Data Set Ready).
NOCTS	nocts	14	Unbenutzt.
DETECTEDBREAK	detectedBreak	15	Letztes Zeichen war ein Terminator.

IOExtSer

Offset		C (devices/serial.h)	Modula-2 (Serial)
Hex	Dez	struct IOExtSer	IOExtSer =
		{	RECORD
0000	0	struct IOStdReq IOSer;	ioSer:IOStdReq;
0030	48	ULONG io_CtlChar;	ctlChar:LONGCARD;
0034	52	ULONG io_RBufLen;	rBufLen:LONGCARD;
0038	56	ULONG io_ExtFlags;	extFlags:ExtSerFlagSet;
003C	60	ULONG io_Baud;	baud:LONGCARD;
0040	64	ULONG io_BrkTime;	brkTime:LONGCARD;
0044	68	struct IOTArray io_TermArray;	termArray:IOTArray;
004C	76	UBYTE io_ReadLen;	readLen:UByte;
004D	77	UBYTE io_WriteLen;	writeLen:UByte;
004E	78	UBYTE io_StopBits;	stopBits:UByte;
004F	79	UBYTE io_SerFlags;	serFlags:SerFlagSet;
0050	80	UWORD io_Status;	status:StatusSet;
0052	82	};	END;

Assembler Syntax: Strukturname = *IOEXTSER*

Definition der Parameter in Versalien.

Datenstrukturenreferenz Keine.

Routinenreferenz »Exec«: *OpenDevice.*

Beschreibung

Bei der *IOExtSer*-Datenstruktur handelt es sich um eine für das Serial-Device erweiterte *IOStdReq*-Datenstruktur. Für den Datenaustausch stehen insbesondere die Standard-Befehle der *IOStdReq*-Datenstruktur zur Verfügung.

❑ **IOSer 0x0000 (0)**

Eine eingebundene *IOStdReq*-Datenstruktur, über die die Kommunikation mit dem Device abläuft. Eine Beschreibung der einzelnen Befehle steht am Anfang dieses Abschnitts.

8

- **io_CtlChar 0x0030 (48)** In diesem 4 Byte langen Feld sind bis zu vier Kontrollzeichen festgelegt. Unterstützt werden zur Zeit nur die Protokollzeichen xON und xOFF. Die Bits 16–23 definieren das xOFF-, die Bits 24–31 das xON-Zeichen. Voreingestellt sind für xON 0x11 (17) und für xOFF 0x13 (19).
- **io_RBufLen 0x0034 (52)** Die Länge des Lesepuffers in Bytes (normalerweise 512 Byte).
- **io_ExtFlags 0x0038 (56)** Eine zur Zeit noch nicht unterstützte 4-Byte-Marke für Zusatzinformationen.
- **io_Baud 0x003C (60)** Die Baudrate für die Übertragung.
- **io_BrkTime 0x0040 (64)** Die Zeitdauer, die ein Abbruchsignal haben muß, damit es als solches erkannt wird. Diese Angabe erfolgt in Mikrosekunden.
- **io_TermArray 0x0044 (68)** Eine eingebundene *IOTArray*-Datenstruktur, in der bis zu acht Terminatoren definiert sind.
- **io_ReadLen 0x004C (76)** Die Anzahl der Bits, die beim Lesen ein Zeichen definieren. Normalerweise also 7 oder 8.
- **io_WriteLen 0x004D (77)** Die Anzahl der Bits, die beim Schreiben ein Zeichen definieren. Normalerweise also 7 oder 8.
- **io_StopBits 0x004E (78)** Die Anzahl der Stopbits, normalerweise 0, 1 oder 2.
- **io_SerFlags 0x004F (79)** Die Bits in dieser 1-Byte-Marke haben gesetzt folgende Bedeutung (zu den Definitionen in C gehört das Präfix: *SERF_*; in Assembler werden die Bits direkt definiert: *BITDEF SER*,<Name>,<Bit>;):

C	Modula-2	Bit	Beschreibung
PARTY_ON	partyOn	0	Paritätsüberprüfung ist eingeschaltet.
PARTY_ODD	partyOdd	1	Bit gesetzt: Es wird auf ungerade Parität geprüft. Bit gelöscht: Es wird auf gerade Parität geprüft.
7WIRE	sevenWire	2	Neben den drei Standard-Übertragungsleitungen TXD (Transmit Data), RXD (Receive Data) und GND (Ground) stehen noch folgende zur Verfügung: RTS (Request To Send), CTS (Clear To Send), DSR (Data Set Ready) und DCD (Data Carrier Detect).

C	Modula-2	Bit	Beschreibung
QUEUEDBRK	queuedBrk	3	Das Break-Kommando wird nicht sofort, sondern erst nach Beendigung des laufenden Schreib-Lese-Vorgangs gesendet.
RAD_BOOGIE	radBoogie	4	Steigerung der Lesegeschwindigkeit durch: 1. Weglassen der Paritätskontrolle; 2. Umgehung des xON-/xOFF Protokolls und 3. 8 Bit pro Zeichen setzen.
SHARED	shared	5	Die Schnittstelle ist auch von anderen Tasks aus zugängig.
EOFMODE	eofMode	6	Anstelle des Standard-Abbruchsignals (0) sind bis zu acht Terminatoren in *io_TermArray* definiert.
XDISABLED	xDisabled	7	xON-/xOff – Protokoll ist abgeschaltet.

❑ **io_Status 0x0050 (80)**

In diesem Langwort steht nach der Anwendung des Befehls *PDCMD_QUERY* der aktuelle Status der seriellen Schnittstelle bzw. des angeschlossenen Geräts. In C und Assembler sind nur die Werte des höheren Wortes extra definiert. Dazu gehört in C das Präfix: *IO_STATF_*; in Assembler werden die Bits direkt definiert: *BITDEF IOST*,<Name>,<Bit>;

C	Modula-2	Bit	Beschreibung
	dataSetReady	3	Bit gelöscht: Schnittstelle im DSR-Zustand.
	clearToSend	4	Bit gelöscht: Schnittstelle im CTS-Zustand.
	carrierDetect	5	Bit gelöscht: Das Trägersignal ist vorhanden.
	readyToSend	6	Bit gelöscht: Schnittstelle im RTS-Zustand.
	dataTerminalReady	7	Bit gelöscht: Schnittstelle im DTR-Zustand.
OVERRUN	overrun	8	Bit gesetzt: Der Lesepuffer ist voll.
WROTEBREAK	wroteBreak	9	Bit gesetzt: Ein BRK-Signal wurde gesendet.
READBREAK	readBreak	10	Bit gesetzt: Ein BRK-Signal wurde gelesen.
XOFFWRITE	xOffWrite	11	Bit gesetzt: Ein xOFF-Signal wurde gesendet.

C	Modula-2	Bit	Beschreibung
XOFFREAD	xOffRead	12	Bit gesetzt: Ein xOFF-Signal wurde gelesen.

IOTArray

Offset		C (devices/serial.h)	Modula-2 (Serial)
Hex	Dez	struct IOTArray	IOTArray =
		{	RECORD
0000	0	ULONG TermArray0;	termArray0:LONGCARD;
0004	4	ULONG TermArray1;	termArray1:LONGCARD;
0008	8	};	END;

Assembler Syntax: Strukturname = *TERMARRAY*
TermArray0 —> *TERMARRAY_0*
TermArray1 —> *TERMARRAY_1*

Datenstrukturenreferenz »Serial«: *IOExtSer.*

Routinenreferenz Keine.

Beschreibung

In dieser Datenstruktur können bis zu acht Terminatoren definiert werden, bei deren Auftreten sofort der Schreib-Lese-Vorgang abgebrochen wird. Werden weniger als acht Terminatoren definiert, so müssen die freien Stellen mit dem Wert des letzten Terminators belegt werden.

❑ **TermArray0 0x0000 (0)** In diesem 4 Byte langen Wert werden die ersten vier Terminatoren definiert.

❑ **TermArray1 0x0004 (4)** In diesem 4 Byte langen Wert werden die letzten vier Terminatoren definiert.

8

8.11 Das Timer-Device

Dieses Device ermöglicht die Benutzung der beiden Timer des Amiga. Es stehen zwei Units, die sich in der Quelle ihres Taktes unterscheiden, zur Verfügung. Der eine benutzt das »vertikal blank interrupt«, während der andere den Timer des CIA-CHIPS gebraucht. Diese unterscheiden sich voneinander in der Auflösung (kleinste Zeiteinheit) und der Genauigkeit. Der CIA-Timer gibt Zeiträume in Mikrosekunden an, wird aber ab einer Sekunde bereits ungenau. Der Vertikal-Blank-Timer mißt auch über längere Zeiträume genau, hat aber nur eine Genauigkeit bis zu einer 50stel-Sekunde. Je nach Anwendung wird man sich also für den einen oder anderen Timer entscheiden. Ein Timer-Device wird durch den Aufruf von *OpenDevice* mit einer entsprechend initialisierten *timerequest*-Datenstruktur geöffnet. Als Unit können Sie, je nach dem, welchen Timer Sie benutzen wollen, einen der folgenden Werte übergeben:

C	Modula-2	Dez	Bedeutung
UNIT_MICROHZ	microHz	0	CIA-Timer.
UNIT_VBLANK	vBlank	1	Vertical Blank Timer.

timerRequest

Offset		C (devices/timer.h)	Modula-2 (Timer)
Hex	Dez	struct timerequest	Timerequest =
		{	RECORD
0000	0	struct IORequest tr_node;	node:IORequest;
0020	32	struct timeval tr_time;	time:TimeVal
0028	40	};	END;

Assembler Syntax: Strukturname = *TIMEREQUEST*
tr_Node —> IOTV_TIME
tr_time —> IOTV_SIZE

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

8

Beschreibung

Diese Datenstruktur ist eine erweiterte *IORequest*-Datenstruktur. Sie dient zum Öffnen und zur Ansteuerung des Timer-Devices.

❑ **tr_Node 0x0000 (0)** Eine *IORequest*-Datenstruktur, die zur Ansteuerung des Timer-Devices verwendet wird. Es können mittels *DoIO* und *SendIO* folgende Befehle an das Device übermittelt werden:

C	Modula2	Dez	Bedeutung
TR_ADDREQUEST	addRequest	9	Warten, bis die in *tr_time* angegebene Zeit verstrichen ist.
TR_GETSYSTIME	getSysTime	10	Die Systemzeit in *tr_Time* einlesen.
TR_SETSYSTIME	setSysTime	11	Die Systemzeit auf die Zeit in *tr_Time* setzen.

❑ **tr_time 0x0020 (32)** Dies ist eine *timeval*-Datenstruktur, in der Zeitwerte an das Timer-Device übergeben werden.

timeval

Offset		C (devices/timer.h)	Modula-2 (Timer)
Hex	Dez	struct timeval	TimeVal =
		{	RECORD
0000	0	ULONG tv_secs;	secs:LONGCARD;
0004	4	ULONG tv_micro;	micro:LONGCARD;
0008	8	};	END;

Assembler Synatax: Strukturname = *TIMEVAL*
Definition der Parameter in Versalien.

Datenstrukturenreferenz »Input«: *InputEvent.*
»Intuition«: *Preferences.*
»Timer«: *IOTimer, timerRequest.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur dient zur Speicherung einer Zeit in Form von Mikrosekunden und Sekunden. Bei der Angabe der Systemzeit wird die Anzahl der Sekunden seit dem 1.1.1978 gezählt.

- **tv_secs 0x0000 (0)** Hier finden Sie die Anzahl der Sekunden.
- **tv_micro 0x0004 (4)** Dieses Feld gibt die Anzahl der Mikrosekunden an.

8

8.12 Das TrackDisk-Device

Dieses Device erlaubt den direkten Zugriff auf die Tracks und Blöcke der Diskette. Sie wird durch einen Aufruf der *OpenDevice*-Routine mit einer *IOExtTD*-Datenstruktur geöffnet. Als Unit wird an *OpenDevice* die Nummer des Laufwerks, das durch dieses Device angesteuert wird (0 bis 4), übergeben.

Folgende Standardbefehle stehen im TrackDisk-Device zur Verfügung:

C	Modula-2	Dez	Beschreibung
CMD_READ	read	2	Einen oder mehrere Blocks von Diskette lesen.

iotd_Req.io_Offset: Die Blocknummer (0..1759) * 512 (Bytes per Block).
iotd_Req.io_Data: Ein Adreßzeiger auf einen Puffer im Chip-RAM der Größe 512 * Blockanzahl, in den die Daten eingelesen werden.
iotd_Req.io_Length: 512 * Anzahl der zu lesenden Blocks.
iotd_SecLabel: Ein Adreßzeiger auf einen Puffer der Länge 16, in den der Sektor-Label eingelesen werden kann, oder *NULL.*

C	Modula-2	Dez	Beschreibung
CMD_WRITE	write	3	Einen oder mehrere Blocks auf Diskette schreiben.

iotd_Req.io_Offset: Die Blocknummer (0..1759) * 512 (Bytes per Block).
iotd_Req.io_Data: Ein Adreßzeiger auf einen Puffer der Größe 512 * Blockanzahl im Chip-RAM, aus dem die Daten gelesen werden.
iotd_Req.io_Length: 512 * Anzahl der zu schreibenden Blocks.
iotd_SecLabel: Ein Adreßzeiger auf einen Puffer der Länge 16, aus dem der Sektor-Label gelesen werden kann, oder *NULL*.

CMD_UPDATE	update	4	Den internen Puffer der Trackdisk-Device auf Diskette schreiben.

Zusätzlich sind zu diesem Device folgende Befehle definiert:

TD_MOTOR	motor	9	Den Laufwerksmotor an-/ausschalten.

iotd_Req.io_Length: 0 = Motor ausschalten; 1 = Motor einschalten.
iotd_Req.io_Actual: Beinhaltet nach dem Aufruf den vorherigen Zustand.

8

TD_SEEK	seek	10	Der zum Track gehörende Laufwerkskopf.

iotd_Req.io_Offset: Die Nummer des Tracks, zu dem der Kopf bewegt werden soll.

TD_FORMAT	format	11	Tracks formatieren. Dabei wird der Inhalt des Puffers direkt auf Diskette geschrieben.

iotd_Req.io_Data: Ein Adreßzeiger auf einen Puffer (im Chip-RAM), der die Länge 512 *11* Track-Anzahl hat. Der Inhalt dieses Puffers wird beim Formatieren auf die Diskette geschrieben.
iotd_Req.io_Length: Die Anzahl der Tracks * 512 (Bytes per Block) * 11 (Sectors per Track)
iotd_Req.io_Offset: Die Nummer des Anfangs-Tracks (0..160).

TD_REMOVE	remove	12	Installiere Diskettenwechselinterrupt

iotd_Req.io_Data: Ein Adreßzeiger auf eine *Interrupt*-Datenstruktur.

C	Modula-2	Dez	Beschreibung
TD_CHANGENUM	changenum	13	Liest die aktuelle Anzahl der Diskettenwechsel.

iotd_Req.io_Actual: Ein Adreßzeiger auf eine *Interrupt*-Datenstruktur.

TD_CHANGESTATE	changeNum	14	Prüfen, ob eine Diskette eingelegt ist.

iotd_Req.io_Actual: 0 = Eine Diskette im Laufwerk eingelegt, sonst keine Diskette im Laufwerk.

TD_PROTSTATUS	protStatus	15	Prüfen, ob Diskette schreibgeschützt ist.

iotd_Req.io_Actual: 0 = Die Diskette ist nicht schreibgeschützt, sonst ist sie schreibgeschützt.

TD_RAWREAD	rawRead	16	Einen bzw. mehrere Tracks byteweise (ohne die DOS-Diskettenstruktur zu beachten) einlesen.

iotd_Req.io_Offset: Die Nummer des ersten zu lesenden Tracks (0..160).
iotd_Req.io_Data: Einen Adreßzeiger auf einen Puffer im Chip-RAM, in den die Daten eingelesen werden. Seine Größe muß 14716 (Bytes per Track) * Track-Anzahl sein.
iotd_Req.io_Length: Hier steht 14716 * Track-Anzahl.
iotd_Req.io_Flags: Hier muß zur Synchronisation des Lesezugriffs dsa Flag *IOTD_INDEXSYNC* gesetzt sein.
iotd_Req.io_Actual: Nach dem Aufruf steht hier die Anzahl der tatsächlich gelesenen Tracks.

TD_RAWWRITE	rawWrite	17	Einen bzw. mehrere Tracks byteweise (ohne die DOS-Diskettenstruktur zu beachten) schreiben.

iotd_Req.io_Offset: Die Nummer des ersten zu schreibenden Tracks (0..160)
iotd_Req.io_Data: Ein Adreßzeiger auf einen Puffer im Chip-RAM, aus dem die Daten gelesen werden. Seine Größe muß 14716 (Bytes per Track) * Track-Anzahl sein.
iotd_Req.io_Length: Hier steht 14716 * Track-Anzahl.
iotd_Req.io_Flags: Hier muß zur Synchronisation des Schreibzugriffs das Flag *IOTD_INDEXSYNC* gesetzt sein.
iotd_Req.io_Actual: Nach dem Aufruf steht hier die Anzahl der tatsächlich geschriebenen Tracks.

C	Modula-2	Dez	Beschreibung
TD_GETDRIVETYPE	getDriveType	18	Laufwerkstyp holen.

iotd_Req.io_Actual: 1 = 3,5-Zoll-Laufwerk; 2 = 5,25-Zoll-Laufwerk.

TD_GETNUMTRACKS	getNumTracks	19	Die maximale Track-Anzahl eines Laufwerks holen.

iotd_Req.io_Actual: Hier finden Sie nach dem Aufruf die Track-Anzahl (=Zylinderzahl * 2).

TD_ADDCHANGEINT	addChangeInt	20	Installiere Diskettenwechsel-interrupt.Ist robuster als TD_REMOVE.

iotd_Req.io_Data: Ein Adreßzeiger auf eine *Interrupt*-Datenstruktur.

TD_REMCHANGEINT	remChangeInt	21	Entferne einen Diskettenwechsel-interrupt

8

iotd_Req.io_Data: Ein Adreßzeiger auf eine *Interrupt*-Datenstruktur.

TD_LASTCOMM	lastComm	22	Gibt den zuletzt ausgeführten Befehl an.

iotd_Req.io_Actual: Nach dem Aufruf finden Sie hier den gesuchten letzten Befehl.

Die folgenden Befehle haben die gleiche Funktion wie die entsprechenden *TD_* - bzw. *CMD_*-Befehle. Vor Ausführung des Befehls wird aber die Anzahl der Diskettenwechsel geprüft. Ist sie größer als die in *iotd_Count*, so wird der Befehl nicht ausgeführt, da angenommen wird, daß die Diskette gewechselt wurde. Somit wird verhindert, daß versehentlich auf die falsche Diskette zugegriffen wird.

C	Modula-2	Hex	Dez
ETD_READ	extRead	8002	32770
ETD_WRITE	extWrite	8003	32771
ETD_UPDATE	extUpdate	8004	32772
ETD_CLEAR	extClear	8005	32773
ETD_MOTOR	extMotor	8009	32777
ETD_SEEK	extSeek	800A	32779
ETD_FORMAT	extFormat	800B	32780

C	Modula-2	Hex	Dez
ETD_RAWREAD	extRawRead	8010	32784
ETD_RAWWRITE	extRawWrite	8011	32785

Folgende Fehlermeldungen sind für das TrackDisk-Device definiert (zu den Definitionen in C und Assembler gehört das Präfix *TDERR_*):

C	Modula-2	Dez	Beschreibung
NOTSPECIFIED	notSpecified	20	Ein unidentifizierter Fehler.
NOSECHDR	noSecHdr	21	Es wurde kein Sektor-Header gefunden.
BADSECPREAMBLE	badSecPreamble	22	Ein Fehler im Sektor-Preamble.
BADSECID	badSecID	23	Die Sektor ID ist nicht in Ordnung.
BADHDRSUM	badHdrSum	24	Die Checksumme des Sektor-Headers ist falsch.
BADSECSUM	badSecSum	25	Die Checksumme des Sektors stimmt nicht.
TOOFEWSECS	tooFewSecs	26	Der Track hat zu wenig Sektoren.
BADSECHDR	badSecHdr	27	Der Sektor-Header ist nicht in Ordnung.
WRITEPROT	writeProt	28	Die Diskette ist schreibgeschützt.
DISKCHANGED	diskChanged	29	Die Diskette wurde gewechselt.
SEEKERROR	seekError	30	Ein Fehler beim Positionieren des Schreib-Lese-Kopfes.
NOMEM	noMem	31	Nicht genug Speicherplatz vorhanden.
BADUNITNUM	badUnitNum	32	Die Laufwerksnummer stimmt nicht.
BADDRIVETYPE	badDriveType	33	Der Laufwerkstyp stimmt nicht.
DRIVEINUSE	driveInUse	34	Es wird bereits auf das Laufwerk zugegriffen.
POSTRESET	postReset	35	???

8

IOExtTD

Offset		C (devices/trackdisk.h)	Modula-2 (TrackDisk)
Hex	Dez	struct IOExtTD {	IOExtTD = RECORD
0000	0	struct IOStdReq iotd_Req;	req:IOStdReq;
0030	48	ULONG iotd_Count;	count:LONGCARD;

Offset		C (devices/trackdisk.h)	Modula-2 (TrackDisk)
Hex	Dez		
0034	52	ULONG iotd_SecLabel;	secLabel:LONGCARD;
0038	56	};	END;

Assembler Syntax: Strukturname = *IOEXTTD*
Definition der Parameter in Versalien.

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Dies ist eine erweiterte *IOStdReq*-Datenstruktur, die zum Öffnen und zur Ansteuerung des Trackdisk-Devices dient.

❑ **iotd_Req 0x0000 (0)** Eine *IOStdReq*-Datenstruktur, in der Befehle, Fehlermeldungen und Daten an und von dem Trackdisk-Device übermittelt werden. Die Befehle, die am Anfang des Kapitels beschrieben wurden, stehen im Command-, die Fehlermeldungen im Error-Feld. Das Data-Feld beinhaltet einen Adreßzeiger auf den Datenbuffer. Seine Größe hängt vom Befehl ab, er muß sich aber im Chip-RAM befinden. Das Unit-Feld zeigt auf eine *TDU_PublicUnit*-Datenstruktur.

❑ **iotd_Count 0x0030 (48)** Hier steht die aktuelle Anzahl der Diskettenwechsel. Sie wird von den *ETD_*-Befehlen (siehe oben) benötigt.

❑ **iotd_SecLabel 0x0034 (52)** Hier kann ein Adreßzeiger auf einen 16 Byte großen Puffer für den Sektor-Label stehen.

TDUPublicUnit

Offset		C (devices/trackdisk.h)	Modula-2 (TrackDisk)
Hex	Dez	struct TDU_PublicUnit	TDUPublicUnit =
		{	RECORD
0000	0	struct Unit tdu_Unit;	unit:Unit;
0026	38	UWORD tdu_Comp01Track;	comp01Track:CARDINAL;
0028	40	UWORD tdu_Comp10Track;	comp10Track:CARDINAL;
002A	42	UWORD tdu_Comp11Track;	comp11Track:CARDINAL;
002C	44	ULONG tdu_StepDelay;	stepDelay:LONGCARD;

Offset		C (devices/trackdisk.h)	Modula-2 (TrackDisk)
Hex	Dez		
0030	48	ULONG tdu_SettleDelay;	settleDelay:LONGCARD;
0034	52	UBYTE tdu_RetryCnt;	retryCnt:[0..255]
0036	54	};	END;

Assembler Syntax: Strukturname = *TDUPUBLICUNIT*
Definition der Parameter in Versalien.

Datenstrukturenreferenz »Trackdisk«: *IOExtTD*.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur beschreibt ein Trackdisk-Device-Unit.

- ❑ **tdu_Unit 0x0000 (0)** Eine eingebundene *Unit*-Datenstruktur, die unter anderem die *MsgPort*-Datenstruktur des Devices enthält.
- ❑ **tdu_Comp01Track 0x0026 (38)** ???
- ❑ **tdu_Comp10Track 0x0028 (40)** ???
- ❑ **tdu_Comp11Track 0x002A (42)** ???
- ❑ **tdu_StepDelay 0x002C (44)** Anscheinend die Zeit in Mikrosekunden, die nach jedem Schritt des Steppermotors gewartet werden muß.
- ❑ **tdu_SettleDelay 0x0030 (48)** Ist wohl die Zeit, die nach dem *SEEK*-Befehl gewartet werden muß.
- ❑ **tdu_RetryCnt 0x0034 (52)** Hier steht die Anzahl von Versuchen, die unternommen werden, bevor ein Fehler gemeldet wird.

8

8.13 Andere Datenstrukturen der Devices

Bei den folgenden Datenstrukturen handelt es sich um Erweiterungen zu verschiedenen Devices.

BadBlockBlock

Offset		C (devices/hardblocks.h)	Modula-2 (Handbuch)
Hex	Dez	struct BadBlockBlock	BadBlockBlock =
		{	RECORD
0000	0	ULONG bbb_ID;	id: ARRAY [0..3] OF CHAR;

Offset		C (devices/hardblocks.h)	Modula-2 (HardBlocks)
Hex	Dez		
0004	4	ULONG bbb_SummedLongs;	summedLongs: LONGCARD;
0008	8	ULONG bbb_ChkSum;	chkSum: LONGINT;
000C	12	ULONG bbb_HostID;	hostID: LONGCARD;
0010	16	ULONG bbb_Next;	next: LONGCARD;
0014	20	ULONG bbb_Reserved;	reserved: LONGCARD;
0018	24	struct BadBlockEntry	blockPairs: ARRAY
		bbb_BadBlockPairs[61]	[0..60] OF BadBlockEntry
0200	512	}	END;

Datenstrukturenreferenz »Hardblocks«: *RigidDiskBlock*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur gehört zu den neueren und ist noch nicht bei allen Compilern in den Include-Files enthalten. Sie definiert den Kopf einer Liste der kaputten Blocks einer Festplatte. Die Nummer der defekten Blocks sowie die der Ausweichblocks werden in einem Feld von *BadBlockEntry*-Datenstrukturen gespeichert. Die *BadBlockBlock*-Datenstruktur samt Liste bildet auf Festplatte normalerweise einen Block. Falls mehr als 61 kaputte Blocks vorhanden sind, können mehrere solcher Blocks über das *bbb_Next*-Feld verkettet werden.

8

- **bbb_ID 0x0000 (0)** Eine vier Zeichen lange Kennung, die den Wert »BADB« haben muß.
- **bbb_SummedLongs 0x0004 (4)** Die Anzahl der Bytes dieser Datenstruktur, über die die Checksumme gebildet wird.
- **bbb_ChkSum 0x0008 (8)** Die Checksumme des Blocks.
- **bbb_HostID 0x000C (12)** Die SCSI-Zielkennung.
- **bbb_Next 0x0010 (16)** Die Nummer des nächsten Blocks in der Liste der *BadBlock-Block*-Datenstrukturen.
- **bbb_Reserved 0x0014 (20)** Für künftige Erweiterungen.
- **bbb_BadBlockPairs 0x0018 (24)** Eine Liste von 61 *BadBlockEntry*-Datenstrukturen, von denen jede einen kaputten Block und den dazugehörenden Ersatzblock beschreibt.

BadBlockEntry

Offset		C (devices/hardblocks.h)	Modula-2 (HardBlocks)
Hex	Dez	struct BadBlockEntry	BadBlockEntry =
		{	RECORD
0000	0	ULONG bbe_BadBlock;	badBlock: LONGCARD;
0004	4	ULONG bbe_GoodBlock;	goodBlock: LONGCARD;
0008	8	}	END;

Datenstrukturenreferenz »Hardblocks«: *BadBlockBlock.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur gehört zu den neueren und ist noch nicht bei allen Compilern in den Include-Files enthalten. Sie gibt die Nummer eines kaputten Blocks und des dazugehörenden Ersatzblocks an. Mit Hilfe solcher Datenstrukturen wird eine Liste aller Defekte einer Festplatte erstellt (siehe *BadBlocksBlock*).

❑ **bbe_BadBlock 0x0000 (0)** Die Nummer des kaputten Blocks.

❑ **bbe_GoodBlock 0x0004 (4)** Die Nummer des zu *bbe_BadBlock* gehörenden intakten Ersatzblocks.

BootBlock

8

Offset		C (devices/bootblock.h)	Modula-2 (BootBlock)
Hex	Dez	struct BootBlock	BootBlock =
		{	RECORD
0000	0	UBYTE bb_id[4];	id:ARRAY [0..3] OF CHAR;
0004	4	LONG bb_chksum;	chkSum:LONGINT;
0008	8	LONG bb_dosblock;	dosBlock:LONGINT;
000C	12	};	END;

Assembler Syntax: Strukturname = *BB*

Definitionen der einzelnen Parameter in Versalien.

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur gibt den Aufbau eines Bootblocks wieder bzw. wird dazu verwendet, einen solchen aufzubauen. Hinter dem letzten Feld kann der Code des Bootprogramms stehen.

❑ **bb_id 0x0000 (0)**

Die hier stehende Zeichenkette gibt an, um was für eine Diskette es sich handelt. Es sind die folgenden zwei Werte zugelassen:

C	Modula-2	Wert	Bedeutung
BBNAME_DOS	nameDos	"DOS\0"	Dies ist eine Amiga-DOS-Diskette.
BBNAME_KICK	nameKick	"KICK"	Dies ist eine Kickstart-Diskette.

❑ **bb_chksum 0x0004 (4)**

In diesem Feld steht die Checksumme des gesamten (inklusive Bootcode) Bootblocks.

❑ **bb_dosblock 0x0008 (8)**

Hier finden Sie die Nummer des Rootblocks der Diskette.

ConUnit

Offset Hex	Dez	C (devices/conunit.h)	Modula-2 (ConUnit)
		struct ConUnit	ConUnit =
		{	RECORD
0000	0	struct MsgPort cu_MP;	mp:MsgPort;
0022	34	struct Window *cu_Window;	window:WindowPtr;
0026	38	WORD cu_XCP;	xCP:INTEGER;
0028	40	WORD cu_YCP;	yCP:INTEGER;
002A	42	WORD cu_XMax;	xMax:INTEGER;
002C	44	WORD cu_YMax;	yMax:INTEGER;
002E	46	WORD cu_XRSize;	xRSize:INTEGER;
0030	48	WORD cu_YRSize;	yRSize:INTEGER;
0032	50	WORD cu_XROrigin;	xROrigin:INTEGER;
0034	52	WORD cu_YROrigin;	yROrigin:INTEGER;
0036	54	WORD cu_XRExtant;	xRExtant:INTEGER;
0038	56	WORD cu_YRExtant;	yRExtant:INTEGER;
003A	58	WORD cu_XMinShrink;	xMinShrink:INTEGER;
003C	60	WORD cu_YMinShrink;	yMinShrink:INTEGER;
003E	62	WORD cu_XCCP;	xcCP:INTEGER;
0040	64	WORD cu_YCCP;	ycCP:INTEGER;
0042	66	struct KeyMap cu_KeyMapStruct;	keyMapStruct:KeyMap;
0062	98	UWORD cu_TabStops[80];	tabStops:ARRAY [0..79] OF CARDINAL;
0102	258	BYTE cu_Mask;	mask:UByte;
0103	259	BYTE cu_FgPen;	fgPen:UByte;
0104	260	BYTE cu_BgPen;	bgPen:UByte;
0105	261	BYTE cu_AOLPen;	aolPen:UByte;
0106	262	BYTE cu_DrawMode;	drawMode:DrawModeSet;

8

Offset		C (devices/conunit.h)	Modula-2 (ConUnit)
Hex	Dez		
0107	263	BYTE cu_AreaPtSz;	areaPtSz:UByte;
0108	264	APTR cu_AreaPtrn;	areaPtrn:ADDRESS;
010C	268	UBYTE cu_Minterms[8];	minTerms:ARRAY [0..7] OF UByte;
0114	276	struct TextFont *cu_Font;	font:TextFontPtr;
0118	280	UBYTE cu_AlgoStyle;	algoStyle:UByte;
0119	281	UBYTE cu_TxFlags;	txFlags:UByte;
011A	282	UWORD cu_TxHeight;	txHeight:CARDINAL;
011C	284	UWORD cu_TxWidth;	txWidth:CARDINAL;
011E	286	UWORD cu_TxBaseline;	txBaseLine:CARDINAL;
0120	288	UWORD cu_TxSpacing;	txSpacing:CARDINAL;
0122	290	UBYTE cu_Modes[3];	modes:ARRAY [0..2] OF UByte;
0125	293	UBYTE cu_RawEvents[3];	rawEvents:ARRAY [0..2] OF UByte;
0128	296	};	END;

Datenstrukturenreferenz »Exec«: *IOStdReq.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur bildet die Schnittstelle zwischen dem Console-Device und dem Ein-/Ausgabe-Fenster. Es wird beim Öffnen eines Console-Devices automatisch initialisiert und ihre Adresse in das Unit-Feld der *IOStdReq*-Datenstruktur eingetragen. Da »Console« über einen eigenen, in der *Window*-Datenstruktur nicht eingetragenen Rastport in das Fenster schreibt, sind die Rastport-bezogenen Felder der Conunit sehr nützlich.

- **cu_MP 0x0000 (0)** — Eine eingebundene *MsgPort*-Datenstruktur, die zu dem Console-Device einen Messageport organisiert.
- **cu_Window 0x0022 (34)** — Hier steht ein Adreßzeiger auf die *Window*-Datenstruktur des Fensters, zu dem das Console-Device gehört.
- **cu_XCP 0x0026 (38)** — Die x-Koordinate des zuletzt ausgegebenen Zeichens (in Zeichen).
- **cu_YCP 0x0028 (40)** — Die y-Koordinate des zuletzt ausgegebenen Zeichens (in Zeichen).
- **cu_XMax 0x002A (42)** — Die größte x-Koordinate, an der ein Zeichen noch ausgegeben werden kann (in Zeichen).
- **cu_YMax 0x002C (44)** — Die größte y-Koordinate, an der ein Zeichen noch ausgegeben werden kann (in Zeichen).
- **cu_XRSize 0x002E (46)** — Die Breite der Zeichenmatrix in Pixel.
- **cu_YRSize 0x0030 (48)** — Die Höhe der Zeichenmatrix in Pixel.

- **cu_XROrigin 0x0032 (50)** Die minimale x-Koordinate in Pixel, an der ein Zeichen ausgegeben werden kann. Normalerweise 0 + Rahmenbreite + 1.
- **cu_YROrigin 0x0034 (52)** Die minimale y-Koordinate in Pixel, an der ein Zeichen ausgegeben werden kann. Normalerweise 0 + Titelleistenbreite + 1.
- **cu_XRExtant 0x0036 (54)** Die x-Koordinate der rechten Grenze, des für die Zeichenausgabe zugelassenen Bereichs in Pixel.
- **cu_YRExtant 0x0038 (56)** Die y-Koordinate der unteren Grenze, des für die Zeichenausgabe zugelassenen Bereichs in Pixel.
- **cu_XMinShrink 0x003A (58)** Die minimale Breite des für die Zeichenausgabe bestimmten Bereichs. Diese Größe hängt von der minimalen Fensterbreite ab.
- **cu_YMinShrink 0x003C (60)** Die minimale Höhe des für die Zeichenausgabe bestimmten Bereichs. Diese Größe hängt von der minimalen Fensterhöhe ab.
- **cu_XCCP 0x003E (62)** Die x-Koordinate der aktuellen Cursorposition in Zeichen.
- **cu_YCCP 0x0040 (64)** Die y-Koordinate der aktuellen Cursorposition in Zeichen.
- **cu_KeyMapStruct 0x0042 (66)** Eine eingebundene *KeyMap*-Datenstruktur, die die von der Console verwendete Tastaturbelegung bestimmt.
- **cu_TabStops 0x0062 (98)** Eine Tabelle, in der die Positionen der Tabulatorstops gespeichert sind.

8

BEACHTE:Die nachfolgenden Felder sind Angaben zu den Eigenschaften des von dem Console-Device benutzten Rastports und sind in ihrer Bedeutung mit den entsprechenden Komponenten der *RastPort*-Datenstruktur identisch. Für die genaue Beschreibung der einzelnen Komponenten und der Bedeutung ihrer Werte siehe »Graphics«.

- **cu_Mask 0x0102 (258)** Die Rastport-Mask für die Zeichenausgabe.
- **cu_FgPen 0x0103 (259)** Die aktuelle Vordergrundfarbe für die Textausgabe durch das Console-Device.
- **cu_BgPen 0x0104 (260)** Die aktuelle Hintergrundfarbe für die Textausgabe durch das Console-Device.
- **cu_AOLPen 0x0105 (261)** Das Feld gibt die Area-Outline-Farbe des Console-Rastports an.
- **cu_DrawMode 0x0106 (262)** Das Feld gibt den Zeichenmodus des Console-Rastports an.
- **cu_AreaPtSz 0x0107 (263)** Die Größe des Füllmusters des Cursors.
- **cu_AreaPtrn 0x0108 (264)** Das für den Cursor zu verwendende Füllmuster.
- **cu_Minterms 0x010C (268)** Die Blitter-Minterms für den Console-Rastport.

- **cu_Font 0x0114 (276)** Ein Adreßzeiger auf eine *TextFont*-Datenstruktur, die den für die Textausgabe über das Console-Device benutzten Zeichensatz definiert.
- **cu_AlgoStyle 0x0118 (280)** Gibt die Schriftart an, die für die Textausgabe über »Console« benutzt wird.
- **cu_TxFlags 0x0119 (281)** Die Flags für die Textausgabe. Die Handhabung ist wie in der *RastPort*-Datenstruktur.
- **cu_TxHeight 0x011A (282)** Gibt die Höhe der Zeichen in Pixel an, die für die Textausgabe über »Console« benutzt werden.
- **cu_TxWidth 0x011C (284)** Gibt die Breite der Zeichen in Pixel an, die für die Textausgabe über »Console« benutzt werden.
- **cu_TxBaseline 0x011E (286)** Handhabung dieses Parameters wie in der *RastPort*-Datenstruktur.
- **cu_TxSpacing 0x0120 (288)** Der Abstand zwischen zwei nachfolgenden Zeichen in Pixel.
- **cu_Modes 0x0122 (290)** Handhabung dieses Parameters wie in der *RastPort*-Datenstruktur.
- **cu_RawEvents 0x0125 (293)** Systeminterne Komponente.

FileSysHeaderBlock

Offset		C (devices/hardblocks.h)	Modula-2 (HardBlocks)
Hex	Dez	struct FileSysHeaderBlock	FileSysHeaderBlock =
		{	RECORD
0000	0	ULONG fhb_ID;	id: ARRAY [0..3] OF CHAR;
0004	4	ULONG fhb_SummedLongs;	summedLongs: LONGCARD;
0008	8	ULONG fhb_ChkSum;	chkSum: LONGINT;
000C	12	ULONG fhb_HostID;	hostID: LONGCARD;
0010	16	ULONG fhb_Next;	next: LONGCARD;
0014	20	ULONG fhb_Flags;	flags: LONGSET;
0018	24	ULONG fhb_Reserved[2];	reserved1: ARRAY [0..1] OF LONGCARD;
0020	32	ULONG fhb_DosType;	dosType: LONGCARD;
0024	36	ULONG fhb Version;	version: LONGCARD;
0028	40	ULONG fhb_PatchFlags;	patchFlags: LONGSET;
002C	44	ULONG fhb Type;	type: LONGCARD;
0030	48	ULONG fhb_Task;	task: LONGCARD;
0034	52	ULONG fhb_Lock;	lock: LONGCARD;
0038	56	ULONG fhb_Handler;	handler: LONGCARD;
003C	60	ULONG fhb_StackSize;	stackSize: LONGCARD;
0040	64	LONG fhb_Priority;	priority: LONGINT;
0044	68	LONG fhb_Startup;	startup: LONGINT;

8

Offset		C (devices/hardblocks.h)	Modula-2 (HardBlocks)
Hex	Dez		
0048	72	LONG fhb_SegListBlocks;	segListBlocks: LONGINT;
004C	76	LONG fhb_GlobVec;	globVec: LONGINT;
0060	80	ULONG fhb_Reserved2[23];	reserved2: ARRAY [0..22] OF LONGCARD;
00AC	172	ULONG fhb_Reserved3[21]	reserved3: ARRAY [0..20] OF LONGCARD;
0100	256	}	END;

Datenstrukturenreferenz »Hardblocks«: *RigidDiskBlock*.

Routinenreferenz Keine.

Beschreibung

Hier wird der Aufbau eines Blocks vereinbart, der die besonderen Anforderungen der Festplatte an das Betriebssystem beschreibt.

- **fhb_ID 0x0000 (0)** Eine vier Zeichen lange Kennung, die den Wert »FSHD« haben muß.
- **fhb_SummedLongs 0x0004 (4)** Die Anzahl der Bytes dieser Datenstruktur, über die die Checksumme gebildet wird.
- **fhb_ChkSum 0x0008 (8)** Die Checksumme des Blocks.
- **fhb_HostID 0x000C (12)** Die SCSI-Zielkennung.
- **fhb_Next 0x0010 (16)** Die Nummer des nächsten Blocks, der eine *FileSysHeader*-Datenstruktur beinhaltet.
- **fhb_Flags 0014 (20)** Einige Filesystem-interne Flags.
- **fhb_Reserved 0x0018 (24)** Wird wohl für einen oder mehrere Zwecke reserviert sein.
- **fhb_DosType 0x001C (28)** Diese Komponente gibt an, unter welchem Filesystem die Partition läuft. Er **muß** mit dem Eintrag in *DosEnvec.de_DosType* übereinstimmen.
- **fhb_Version 0x0024 (36)** Gibt die Versionsnummer an.
- **hb_PatchFlags 0x0028 (40)** Wenn hier ein Bit gesetzt ist, dann wird eines (das der Nummer dieses Bits entsprechende) der nachfolgenden Felder anstelle des Standardfeldes der *DeviceNode*-Datenstruktur für dieses Device benutzt.
- **fhb_Type 0x002C (44)** Der Typ der *DeviceNode*-Datenstruktur. Muß in der vorliegenden Version 0 sein.
- **hb_Task 0x0030 (48)** Entspricht dem *dn_Task*-Feld der *DeviceNode*-Datenstruktur.

8

- **fhb_Lock 0x0034 (52)** Dieser Parameter entspricht dem *dn_Lock*-Feld der *DeviceNode*-Datenstruktur.
- **fhb_Handler 0x0038 (56)** Dieser Parameter entspricht dem *dn_Handler*-Feld der *DeviceNode*-Datenstruktur.
- **fhb_StackSize 0x003C (60)** Dieser Parameter entspricht dem *dn_StackSize*-Feld der *DeviceNode*-Datenstruktur.
- **fhb_Priority 0x0040 (64)** Dieser Parameter entspricht dem *dn_Priority*-Feld der *DeviceNode*-Datenstruktur.
- **hb_Startup 0x0044 (68)** Dieser Parameter entspricht dem *dn_Startup*-Feld der *DeviceNode*-Datenstruktur.
- **fhb_SegListBlocks 0x0048 (72)** Die Nummer des ersten Blocks der Segmentliste des Handlers. Dieser Parameter ist äquivalent zu dem *dn_SegmentList*-Feld der *DeviceNode*-Datenstruktur.
- **fhb_GlobVec 0x004C (76)** Entspricht dem *dn_GlobVec*-Feld der *DeviceNode*-Datenstruktur.
- **fhb_Reserved2 0x0050 (80)** Reservierter Speicherplatz.
- **fhb_Reserved3 0x00AC (172)** Noch ein reservierter Speicherplatz.

LoadSegBlock

Offset Hex	Offset Dez	C (devices/hardblocks.h)	Modula-2 (HardBlocks)
		struct LoadSegBlock	LoadSegBlock =
		{	RECORD
0000	0	ULONG lsb_ID;	id: ARRAY [0..3] OF CHAR;
0004	4	ULONG lsb_SummedLongs;	summedLongs: LONGCARD;
0008	8	ULONG lsb_ChkSum;	chkSum: LONGINT;
000C	12	ULONG lsb_HostID;	hostID: LONGCARD;
0010	16	ULONG lsb_Next;	next: LONGCARD;
0014	20	ULONG lsb_LoadData[123]	loadData: ARRAY [0..122] OF LONGCARD;
0200	512	}	END;

Datenstrukturenreferenz »Hardblocks«: *FileSysHeaderBlock.*

Routinenreferenz Keine.

Beschreibung

Die *LoadSegBlock*-Datenstruktur gehört zu den neueren und ist noch nicht bei allen Compilern in den Include-Files enthalten. Sie legt den Aufbau eines Blocks der Segementliste fest, in der der Code des Handlers gespeichert ist (siehe auch *FielSysHeaderBlock*). Jeder derartige Block beinhaltet neben den

»Verwaltungsdaten« (ID, Checksumme etc.), die Nummer des nächsten Blocks und 123 Langworte des Codes.

- **lsb_ID 0x0000 (0)** Eine vier Zeichen lange Kennung, die den Wert »LSEG« hat.
- **lsb_SummedLongs 0x0004 (4)** Die Anzahl der Bytes dieser Datenstruktur, über die die Checksumme gebildet wird.
- **lsb_ChkSum 0x0008 (8)** Die Checksumme des Blocks.
- **lsb_HostID 0x000C (12)** Die SCSI-Zielkennung.
- **lsb_Next 0x0010 (16)** Hier steht die Nummer des nächsten Blocks in der Liste oder Null, falls dies der letzte Block war.
- **lsb_LoadData 0x0014 (20)** Hier folgen 123 Langworte, die einen Teil des zu ladenden Codes enthalten.

InputEvent

Offset Hex	Offset Dez	C (devices/inputevent.h)	Modula-2 (InputEvent)
		struct InputEvent	InputEvent =
		{	RECORD
0000	0	struct InputEvent	nextEvent:InputEventPtr;
		*ie_NextEvent;	
0004	4	UBYTE ie_Class;	class:Class;
0005	5	UBYTE ie_SubClass;	subClass:Class;
0006	6	UWORD ie_Code;	code:CARDINAL;
0008	8	UWORD ie_Qualifier;	qualifier:QualifierSet;
		union	CASE :INTEGER OF
		{	
		struct	
		{	
000A	10	WORD ie_x;	|0: x: INTEGER;
000C	12	WORD ie_y;	y: INTEGER;
		} ie_xy;	
000A	10	APTR ie_addr;	|1: eventAddress:ADDRESS
		} ie_position;	END;
000E	14	struct timeval ie_TimeStamp;	timeStamp:TimeVal
0016	22	};	END;

Datenstrukturenreferenz »Input«: *InputEvent.*
»Intuition«: *IntuitionBase.*

Routinenreferenz Keine.

Beschreibung

In dieser Datenstruktur wird ein aufgetretenes Ereignis gespeichert. Die Adresse eines solchen Input-Events wird an die Input-Handler übergeben. Treten mehrere Events auf, so werden mehrere *InputEvent*-Datenstrukturen zu einer Liste verknüpft.

- **ie_NextEvent 0x0000 (0)** — Ein Adreßzeiger zur Verkettung mehrerer *InputEvent*-Datenstrukturen. In der so erzeugten Liste muß in der letzten *InputEvent*-Datenstruktur an dieser Stelle der Wert *NULL* stehen.

- **ie_Class 0x0004 (4)** — Dieser Parameter gibt die Art des aufgetretenen Ereignisses an. Folgende Werte sind für die einzelnen Meldungen möglich (zu den Definitionen in C und Assembler gehört das Präfix *IECLASS_*; In Modula-2 sind dies Elemente des Aufzählungstyps *Class*):

C	Modula-2	Dez	Beschreibung
NULL	null	0	Eine »NIX«-Meldung.
RAWKEY	rawkey	1	Meldung einer Tastatureingabe (Raw-Code).
RAWMOUSE	rawmouse	2	Bewegung der Maus (Raw-Daten vom GamePort-Device).
EVENT	event	3	Ein interner Event des Console-Devices.
POINTERPOS	pointerpos	4	Informationen zur Position des Mauspfeils.
TIMER	timer	6	Ein Timer-Event.
GADGETDOWN	gadgetdown	7	Ein Gadget wurde gedrückt, also angeklickt.
GADGETUP	gadgetup	8	Ein Gadget wurde wieder »losgelassen«.
REQUESTER	requester	9	Ein Requester wurde geöffnet.
MENULIST	menulist	10	Ein Menü-Ereignis.
CLOSEWINDOW	closewindow	11	Das Close-Gadget wurde angeklickt.
SIZEWINDOW	sizewindow	12	Das Size-Gadget wurde angeklickt.
REFRESHWINDOW	refreshwindow	13	An dem Fenster, auf dessen Window-Datenstruktur das Feld ie_EventAddress zeigt, muß ein Refresh durchgeführt werden.
NEWPREFS	newprefs	14	Die Preferences-Einstellungen wurden verändert.
DISKREMOVED	diskremoved	15	Eine Diskette wurde aus einem Laufwerk entnommen.
DISKINSERTED	diskinserted	16	Eine Diskette wurde in ein Laufwerk eingelegt.

C	Modula-2	Dez	Beschreibung
ACTIVEWINDOW	activewindow	17	Das zugehörige Fenster wurde aktiviert.
INACTIVEWINDOW	inactivewindow	18	Das zugehörige Fenster wurde inaktiviert.

- **ie_SubClass 0x0005 (5)** In diesem Feld können zusätzliche Informationen übertragen werden.
- **ie_Code 0x0006 (6)** Folgende Konstanten sind für dieses Feld definiert (zu den Definitionen in C und Assembler gehört das Präfix *IECODE_*;

Für *IECLASS_RAWKEY*:

C	Modula-2	Hex	Dez
KEY_CODE_FIRST	keyCodeFirst	00	0
KEY_CODE_LAST	keyCodeLast	77	119
COMM_CODE_FIRST	commCodeFirst	78	120
COMM_CODE_LAST	commCodeLast	7F	127
UP_PREFIX	upPrefix	80	128

8

Für *IECLASS_ANSI*:

C	Modula-2	Hex	Dez
C0_FIRST	c0First	00	0
C0_LAST	c1First	1F	31
ASCII_FIRST	asciiFirst	20	32
ASCII_LAST	asciiLast	7E	126
ASCII_DEL	asciiDel	7F	127
C1_FIRST	c1First	80	128
C1_LAST	c1Last	9F	159
LATIN1_FIRST	latin1First	A0	160
LATIN1_LAST	latin1Last	FF	255

Für *IECLASS_RAWMOUSE*:

C	Modula-2	Hex	Dez	Beschreibung
LBUTTON	lButton	68	104	Event betrifft linke Maustaste.

C	Modula-2	Hex	Dez	Beschreibung
RBUTTON	rButton	69	105	Event betrifft rechte Maustaste.
MBUTTON	mButton	6A	106	Event betrifft beide Maustasten.
NOBUTTON	noButton	FF	255	Event betrifft keine Maustaste.

Für *IECLASS_EVENT*:

C	Modula-2	Hex	Dez	Beschreibung
NEWACTIVE	newActive	01	1	Ein anderes Fenster wurde aktiv.

Für *IECLASS_REQUESTER*:

C	Modula-2	Hex	Dez	Beschreibung
REQCLEAR	reqClear	00	0	Der letzte Requester eines Fensters wurde geschlossen.
REQSET	reqSet	01	1	Wird gesendet, wenn in einem Fenster der erste Requester geöffnet wird.

- **ie_Qualifier 0x0008 (8)**

Der hier stehende Wert gibt bei der Tastatureingabe an, welche Qualifier (<Shift>, <Alt> etc.) gedrückt sind. Folgende Werte sind möglich (zu den Definitionen in C und Assembler gehört das Präfix *IEQUALIFIER_*; In Modula-2 sind dies Elemente des Mengentyps *QualifierSet*):

C	Modula-2	Hex	Beschreibung
LSHIFT	lShift	0001	Linke <Shift>-Taste gedrückt.
RSHIFT	rShift	0002	Rechte <Shift>-Taste gedrückt.
CAPSLOCK	capsLock	0004	<Caps Lock>-Taste gedrückt.
CONTROL	control	0008	<Ctrl>-Taste gedrückt.
LALT	lAlt	0010	Linke <Alt>-Taste gedrückt.
RALT	rAlt	0020	Rechte <Alt>-Taste gedrückt.
LCOMMAND	lCommand	0040	Linke <Amiga>-Taste gedrückt.
RCOMMAND	rCommand	0080	Rechte <Amiga>-Taste gedrückt.
NUMERICPAD	numericPad	0100	Eine Taste des »Zehnerblocks« gedrückt.

C	Modula-2	Hex	Beschreibung
REPEAT	repeat	0200	Tastenwiederholung.
INTERRUPT	interrupt	0400	<Ctrl> + <C>???
MULTIBROADCAST	multiBroadcast	0800	Etwa ein neuer Piratensender???
MIDBUTTOM	midButton	1000	Beide Maustasten gedrückt?
RBUTTON	rightButton	2000	Rechte Maustaste gedrückt.
LEFTBUTTON	leftButton	4000	Linke Maustaste gedrückt.
RELATIVEMOUSE	relativeMouse	8000	Mauskoordinaten sind relativ.

- **ie_x 0x000A (10)** Bei einem Maus-Event steht hier die x-Koordinate des Mauszeigers zum Zeitpunkt des Events.

- **ie_y 0x000C (12)** Bei einem Mausevent steht hier die y-Koordinate des Mauszeigers zum Zeitpunkt des Events.

- **ie_addr 0x000A (10)** Ein Adreßzeiger auf die zum Event gehörende Datenstruktur. Bei Tastatureingaben auf die betreffende *Window*-Datenstruktur, bei Gadget-Events auf die betreffende Gadget-Datenstruktur etc.

- **ie_TimeStamp 0x000E (14)** Ein Adreßzeiger auf eine *TimeVal*-Datenstruktur. In ihr wird die Systemzeit festgehalten, zu der dieser Event aufgetreten ist.

KeyMap

Offset Hex	Dez	C (devices/keymap.h)	Modula-2 (KeyMap)
		struct KeyMap	KeyMap =
		{	RECORD
0000	0	UBYTE *km_LoKeyMapTypes;	loKeyMapTypes:TypesPtr;
0004	4	ULONG *km_LoKeyMap;	loKeyMap:InfoPtr;
0008	8	UBYTE *km_LoCapsable;	loCapsable:BitTablePtr;
000C	12	UBYTE *km_LoRepeatable;	loRepeatable:BitTablePtr;
0010	16	UBYTE *km_HiKeyMapTypes;	hiKeyMapTypes:TypesPtr;
0014	20	ULONG *km_HiKeyMap;	hiKeyMap:InfoPtr;
0018	24	UBYTE *km_HiCapsable;	hiCapsable:BitTablePtr;
001C	28	UBYTE *km_HiRepeatable;	hiRepeatable:BitTablePtr;
0020	32	};	END;

Datenstrukturenreferenz

»ConUnit«:	*ConUnit.*
»Intuition«:	*IntuitionBase, StringInfo.*
»KeyMap«:	*KeyMapNode.*

Routinenreferenz Keine.

Beschreibung

Mit dieser Datenstruktur wird die *Keymap*-Tabelle beschrieben, in der die Umwandlungen der verschiedenen Tastenkombinationen definiert sind. Die Tabelle ist in zwei Teile geteilt, und zwar in eine Lo-Map (Tastencodes von 0x00 bis 0x3F) und eine Hi-Map (Tastencodes von 0x40 bis 0x67).

- **km_LoKeyMapTypes 0x0000 (0)** Ein Adreßzeiger auf eine 64 Byte lange Tabelle (deren Anfangsadresse muß gerade sein!), deren Werte angeben, welche Tastenkombinationen (mit oder ohne Qualifier, wie [Shift], [Alt] und [Ctrl]) für die Tasten der Lo-Map zulässig sind.

C	Modula-2	Hex	Beschreibung
KC_NOQUAL	noQual	0	Für die Taste ist kein Qualifier zugelassen.
KCF_SHIFT	shift	1	Die [Shift]-Tasten sind als Qualifier zugelassen.
KCF_ALT	alt	2	Die [Alt]-Tasten sind als Qualifier zugelassen.
KCF_CONTROL	control	4	Die [Ctrl]-Taste ist als Qualifier zugelassen.
KC_VANILLA	vanilla	7	Kombination von *KCF_SHIFT*, *KCF_ALT* und *CONTROL*.
KCF_STRING	string	40	Es wird kein einzelner ASCII-Code, sondern eine Zeichenkette pro Tastendruck ausgegeben.

BEACHTE: In Modula-2 ist für dieses Feld der Aufzählungstyp *KeyMapTypes* und der Mengentyp *KeyMapTypeSet* definiert. *noQual* und *vanilla* sind natürlich nicht in diesem Aufzählungstyp, sondern als Konstanten definiert. *TypesPtr* ist ein *POINTER TO ARRAY [0..63] OF KeyMapTypeSet*.

- **km_LoKeyMap 0x0004 (4)** Ein Adreßzeiger auf eine Tabelle (deren Anfangsadresse muß gerade sein!) mit 64 *KeyInfo*-Datenstrukturen, in denen für die Tasten der Lo-Map die ASCII-Codes gespeichert sind, die bei Betätigung der jeweiligen Taste ausgegeben werden.

- **km_LoCapsable 0x0008 (8)** Ein Adreßzeiger auf eine 64 Bit lange Tabelle, deren einzelnen Bits zu jeder Taste der Lo-Map festlegen, ob bei betätigter <Caps Lock>-Taste die ge<Shift>ete (TRUE) oder die normale (FALSE) Ausgabe erfolgen soll.

- **km_LoRepeatable 0x000C (12)** Ein Adreßzeiger auf eine 64 Bit lange Tabelle, deren einzelnen Bits zu jeder Taste der Lo-Map festlegen, ob bei gedrückt gehaltener Taste eine automatische Wiederholung erfolgen soll (TRUE) oder nicht (FALSE).

❑ **km_HiKeyMapTypes 0x0010 (16)** — Ein Adreßzeiger auf eine 64 Byte lange Tabelle (deren Anfangsadresse muß gerade sein!), deren Werte angeben, welche Tastenkombinationen (mit oder ohne Qualifier, wie <Shift>, <Alt> und <Ctrl>) für die Tasten der Hi-Map zulässig sind. Für die Tabelle gelten die gleichen Werte wie für *km_LoKeyMapTypes*.

❑ **km_HiKeyMap 0x0014 (20)** — Ein Adreßzeiger auf eine Tabelle (deren Anfangsadresse muß gerade sein!) mit 64 *KeyInfo*-Datenstrukturen, in denen für die Tasten der Hi-Map die ASCII-Codes gespeichert sind, die bei Betätigung der jeweiligen Taste ausgegeben werden.

❑ **km_HiCapsable 0x0018 (24)** — Ein Adreßzeiger auf eine 64 Bit lange Tabelle, deren einzelnen Bits zu jeder Taste der Hi-Map festlegen, ob bei betätigter <Caps Lock>-Taste die »geshiftete« (TRUE) oder die normale (FALSE) Ausgabe erfolgen soll.

❑ **km_HiRepeatable 0x001C (28)** — Ein Adreßzeiger auf eine 64 Bit lange Tabelle, deren einzelnen Bits zu jeder Taste der Hi-Map festlegen, ob bei gedrückt gehaltener Taste eine automatische Wiederholung erfolgen soll (TRUE) oder nicht (FALSE).

KeyMapNode

Offset		C (devices/keymap.h)	Modula-2 (KeyMap)
Hex	**Dez**	struct KeyMapNode	KeyMapNode =
		{	RECORD
0000	0	struct Node kn_Node;	node:Node;
000E	14	struct KeyMap kn_KeyMap;	keyMap:KeyMap;
002E	46	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur ordnet einer *KeyMap*- einer *Node*-Datenstruktur zu, damit über die *KeyMapResource*-Datenstruktur mehrere Keymaps miteinander verknüpft werden können.

❑ **kn_Node 0x0000 (0)** — Eine eingebundene *Node*-Datenstruktur, die vor allem einen Adreßzeiger auf den zu einer Keymap gehörenden Namen beinhaltet.

❑ **kn_KeyMap 0x000E (14)** — Eine eingebundene *KeyMap*-Datenstruktur, durch die eine Keymap definiert ist.

KeyMapResource

Offset		C (devices/keymap.h)	Modula-2 (KeyMap)
Hex	Dez	struct KeyMapResource {	KeyMapResource= RECORD
0000	0	struct Node kr_Node;	node:Node;
000E	14	struct List kr_List;	list:List;
001C	28	};	END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Über diese Datenstruktur können mehrere Keymaps miteinander verknüpft werden.

- **kr_Node 0x0000 (0)** Eine eingebundene *Node*-Datenstruktur zur Verknüpfung der Keymaps.
- **kr_List 0x000E (14)** Durch diese eingebundene *List*-Datenstruktur werden die *KeyMapNode*-Datenstrukturen als eine lineare Liste bereitgehalten.

DeviceData

8

Offset		C (devices/prtbase.h)	Modula-2 (PrtBase)
Hex	Dez	struct DeviceData {	DeviceData = RECORD
0000	0	struct Library dd_Device;	device:Device;
0022	34	APTR dd_Segment;	segment:BPTR;
0026	38	APTR dd_ExecBase;	execBase:ExecBasePtr;
002A	42	APTR dd_CmdVectors;	cmdVectors:ADDRESS;
002E	46	APTR dd_CmdBytes;	cmdBytes:ADDRESS;
0032	50	UWORD dd_NumCommands;	numCommands:CARDINAL;
0034	52	};	END;

Assembler Der Parameter *dd_Device* ist nicht definiert.

Datenstrukturenreferenz »PrtBase«: *PrinterData.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur enthält systeminterne Informationen über das Printer-Device selbst.

- **dd_Device 0x0000 (0)** Eine eingebundene *Library*-Datenstruktur, die das PrtBase-Device in das System einbindet.
- **dd_Segment 0x0022 (34)** Ein Adreßzeiger auf die Segmentliste des Handlers des Printer-Devices.
- **dd_ExecBase 0x0026 (38)** Ein Adreßzeiger auf die *ExecBase*-Datenstruktur.
- **dd_CmdVectors 0x002A (42)** Ein Adreßzeiger auf die Sprungtabelle für die verschiedenen Kommandos, die zu diesem Device definiert sind.
- **dd_CmdBytes 0x002E (46)** Ein Adreßzeiger auf den Speicherbereich, in dem die Kommandos hintereinander gespeichert sind.
- **dd_NumCommands 0x0032 (50)** Hier steht die Anzahl der Kommandos, die in *dd_CmdBytes* gespeichert sind.

PartitionBlock

Offset		C (devices/hardblocks.h)	Modula-2 (HardBlocks)
Hex	**Dez**	struct PartitionBlock	PartitionBlock =
		{	RECORD
0000	0	ULONG pb_ID;	id: ARRAY [0..3] OF CHAR;
0004	4	ULONG pb_SummedLongs;	summedLongs: LONGCARD;
0008	8	LONG pb_ChkSum;	chkSum: LONGINT;
000C	12	ULONG pb_HostID;	hostID: LONGCARD;
0010	16	ULONG pb_Next;	next: LONGCARD;
0014	20	ULONG pb_Flags;	flags: Partition FlagSet;
0018	24	ULONG pb_Reserved1[2];	reserved1: ARRAY [0..1] OF LONGCARD;
0020	32	ULONG pd_DevFlags;	devFlags: LONGSET;
0024	36	UBYTE pb_DriveName[32];	driveName: ARRAY [0..31] OF CHAR;
0044	68	ULONG pb_Reserved2[2];	reserved2: ARRAY [0..14] OF LONGCARD;
004C	76	ULONG pb_Enviroment[17];	enviroment: ARRAY [0..16] OF LONGCARD;
0090	144	ULONG pb_Reserved[15];	reserved: ARRAY [0..14] OF LONGCARD
00CC	204	}	END;

Datenstrukturenreferenz »Hardblocks«: *RigidDiskBlock.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur gehört zu den neueren und ist nicht bei allen Compilern in den Include-Files enthalten. Sie beschreibt einen Block, der eine Partition eines Festplattenlaufwerks identifiziert. Falls die

Festplatte mehrere Partitionen enthält, dann steht die Nummer des ersten solchen *PartitionBlock*-Blocks in der *RigidDiskBlock*-Datenstruktur.

- **pb_ID 0x0000 (0)** — Eine vier Zeichen lange Kennung, die den Wert »PART« haben muß.
- **pb_SummedLongs 0x0004 (4)** — Die Anzahl der Bytes dieser Datenstruktur, über die die Checksumme gebildet wird.
- **pb_ChkSum 0x0008 (8)** — Die Checksumme des Blocks.
- **pb_HostID 0x000C (12)** — Die SCSI-Zielkennung.
- **pb_Next 0x0010 (16)** — Die Nummer des nächsten Blocks in der Liste oder null falls dies der letzte Block ist.
- **pb_Reserved1 0018 (24)** — Reserviert für diverse Zwecke.
- **pd_DevFlags 0x0020 (32)** — Einige private Flags der Partition.
- **pb_DriveName 0x0024 (36)** — Hier finden Sie den Namen der Partition als Zeichenkette.
- **pb_Reserved2 0x0044 (68)** — Reserviert für diverse Zwecke.
- **pb_Enviroment 0x004C (76)** — Eine private Sprung- und Datentabelle der Partition.
- **pb_EReserved 0x0050 (144)** — Reserviert für diverse Zwecke.

PrinterData

Offset		C (devices/prtbase.h)	Modula-2 (PrtBase)
Hex	Dez	struct PrinterData	PrinterData =
		{	RECORD
0000	0	struct DeviceData pd_Device;	device:DeviceData;
0034	52	struct MsgPort pd_Unit;	unit:MsgPort;
0056	86	BPTR pd_PrinterSegment;	printerSegment:BPTR;
005A	90	UWORD pd_PrinterType;	printerType:CARDINAL;
005C	92	struct PrinterSegment *pd_SegmentData;	segmentData:PrinterSegmentPtr;
0060	96	UBYTE *pd_PrintBuf;	printBuf:ADDRESS;
0064	100	int (*pd_PWrite)();	pWrite:PROCEDURE():INTEGER;
0068	104	int (*pd_PBothReady)();	pBothReady:PROCEDURE():INTEGER;
		union {	CASE :INTEGER OF
006C	108	struct IOExtPar pd_p0;	|1: p0:IOParallel;
006C	108	struct IOExtSer pd_s0;	p1:IOParallel;
		} pd_ior0;	
		union {	
00BE	190	struct IOExtPar pd_p1;	|2: s0:IOSerial;
00BE	190	struct IOExtSer pd_s1;	s1:IOSerial;
		} pd_ior1;	END;

Offset		C (devices/prtbase.h)	Modula-2 (PrtBase)
Hex	Dez		
0110	272	struct timerequest pd_TIOR;	tior:IOTimer;
0138	312	struct MsgPort pd_IORPort;	iorPort:MsgPort;
015A	346	struct Task pd_TC;	tc:Task;
01B6	438	UBYTE pd_Stk[2048];	stk:ARRAY [0..2047] OF BYTE;
09B6	2486	UBYTE pd_Flags;	flags:UByte;
09B7	2487	UBYTE pd_pad;	pad:BYTE;
09B8	2488	struct Preferences pd_Preferences;	preferences:Preferences;
0AA0	2720	UBYTE pd_PWaitEnabled;	pWaitEnabled:UByte;
0AA2	2722	};	END;

Assembler Der Parameter *pd_Device* ist nicht definiert.

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur wird von der *ped_Init*-Routine (siehe *PrinterExtendedData*-Datenstruktur) initialisiert.

8

- **pd_Device 0x0000 (0)** Eine eingebundene *DeviceData*-Datenstruktur, die systeminterne Informationen zu dem *PrtBase*-Device enthält.
- **pd_Unit 0x0034 (52)** Eine eingebundene *Unit*-Datenstruktur.
- **pd_PrinterSegment 0x0056 (86)** Ein BPTR auf eine *PrinterSegment*-Datenstruktur.
- **pd_PrinterType 0x005A (90)** Angabe des Druckertyps.
- **pd_SegmentData 0x005C (92)** Ein Adreßzeiger auf die zugehörige *PrinterSegment*-Datenstruktur. Dieser Adreßzeiger steht hier, damit der Druckertreiber mit der Routine *UnLoadSeg* aus dem Speicher entfernt werden kann.
- **pd_PrintBuf 0x0060 (96)** Ein Adreßzeiger auf einen Puffer, der als Zwischenspeicher für die Grafikausgabe benötigt wird.
- **pd_PWrite 0x0064 (100)** Ein Adreßzeiger auf die Ausgabe-Routine, die die Druckdaten über die entsprechende Schnittstelle an den Drucker schickt.
- **pd_PBothReady 0x0068 (104)** Ein Adreßzeiger auf eine Funktion, die am Ende des Druckvorgangs aufgerufen wird (und dann was macht?).
- **pd_pior0 0x006C (108)** Je nachdem, ob der Drucker an der parallelen oder seriellen Schnittstelle angeschlossen ist, ist an dieser Stelle die erste

von zwei *IOExtPar-* oder *IOExtSer*-Datenstrukturen eingebunden.

❑ **pd_piorl 0x00BE (190)**
Je nachdem, ob der Drucker an der parallelen oder seriellen Schnittstelle angeschlossen ist, ist an dieser Stelle die zweite von zwei *IOExtPar-* oder *IOExtSer*-Datenstrukturen eingebunden.

❑ **pd_TIOR 0x0110 (272)**
Eine eingebundene *timerequest*-Datenstruktur, die dem Printer-Device die für Synchronisationen notwendige Zugriffe auf den Timer-Device erlaubt.

❑ **pd_IORPort 0x0138 (312)**
Eine eingebundene *MsgPort*-Datenstruktur. Über diesen Messageport werden die für das Printer-Device bestimmten Nachrichten empfangen.

❑ **pd_TC 0x015A (346)**
Eine eingebundene Task-Datenstruktur, die den Druckvorgang zu einem eigenen Task macht, so daß er im Hintergrund ausgeführt werden kann.

❑ **pd_Stk 0x01B6 (438)**
An dieser Stelle wird für die Datenübertragung ein Stack eingerichtet, der einzelne Zeichen zwischenspeichern kann.

❑ **pd_Flags 0x09B6 (2486)**
Laut [RKA] sind dies die Device-Flags... welche damit auch immer gemeint sind.

❑ **pd_pad 0x09B7 (2487)**
Zwei unbedeutende Füllbytes.

❑ **pd_Preferences 0x09B8 (2488)**
Eine eingebundene *Preferences*-Datenstruktur, die eine Kopie der aktuellen Preferences-Voreinstellungen beinhaltet.

❑ **pd_PWaitEnabled 0x0AA0 (2720)**
Eine systeminterne 1-Byte-Marke. Sie wird jedenfalls im Zusammenhang mit den verschiedenen Wartezuständen des Devices benutzt.

8

PrinterExtendedData

Offset		**C (devices/prtbase.h)**	**Modula-2 (PrtBase)**
Hex	**Dez**	struct PrinterExtendedData	PrinterExtendedData =
		{	RECORD
0000	0	char *ped_PrinterName;	printerName:ADDRESS;
0004	4	VOID (*ped_Init)();	init:PROC;
0008	8	VOID (*ped_Expunge)();	expunge:PROC;
000C	12	int (*ped_Open)();	open:PROCEDURE():INTEGER;
0010	16	VOID (*ped_Close)();	close:PROC;
0014	20	UBYTE ped_PrinterClass;	printerClass:PrinterClassSet;
0015	21	UBYTE ped_ColorClass;	colorClass:ColorClassSet;
0016	22	UBYTE ped_MaxColumns;	maxColumns:UByte;

Offset		C (devices/prtbase.h)	Modula-2 (PrtBase)
Hex	Dez		
0017	23	UBYTE ped_NumCharSets;	numCharSets:UByte;
0018	24	UWORD ped_NumRows;	numRows:CARDINAL;
001A	26	ULONG ped_MaxXDots;	maxXDots:LONGCARD;
001E	30	ULONG ped_MaxYDots;	maxYDots:LONGCARD;
0022	34	UWORD ped_XDotsInch;	xDotsInch:CARDINAL;
0024	36	UWORD ped_YDotsInch;	yDotsInch:CARDINAL;
0026	38	char ***ped_Commands;	commands:ADDRESS;
002A	42	int (*ped_DoSpecial)();	doSpecial:PROCEDURE():INTEGER;
002E	46	int (*ped_Render)();	render:PROCEDURE():INTEGER;
0032	50	LONG ped_TimeoutSecs;	timeoutSecs:LONGINT;
0036	54	char **ped_8BitChars;	eightBitChars:ADDRESS;
003A	58	LONG ped_PrintMode;	printMode:LONGINT;
003E	62	int (*ped_ConvFunc)();	convFunc:PROCEDURE():LONGINT;
0042	66	};	END;

Datenstrukturenreferenz »PrtBase«: *PrinterSegment.*

Routinenreferenz Keine.

Beschreibung

In dieser Datenstruktur werden druckerspezifische Informationen abgelegt.

❑ **ped_PrinterName 0x0000 (0)** Ein Adreßzeiger auf eine Null-terminierte Zeichenkette, in der der Name dieses Druckers (und somit auch des Druckertreibers) steht.

❑ **ped_Init 0x0004 (4)** Ein Adreßzeiger auf eine Initialisierungs-Routine für *PrinterData*-Datenstrukturen, die nach der *LoadSeg*-Routine (Laden des Druckertreibers) aufgerufen wird. Ein Adreßzeiger auf eine *PrinterData*-Datenstruktur muß ihr übergeben werden.

❑ **ped_Expunge 0x0008 (8)** Ein Adreßzeiger auf eine Lösch-Routine für die durch *ped_Init* eingerichtete *PrinterData*-Datenstruktur, die vor der *UnLoadSeg*-Routine aufgerufen wird.

❑ **ped_Open 0x000C (12)** Ein Adreßzeiger auf eine spezifische Open-Routine des Druckertreibers, die unmittelbar nach der *OpenDevice*-Routine aufgerufen wird. Ihr muß ein Adreßzeiger auf die zum PrtBase-Device gehörende *IOStdReq*-Datenstruktur übergeben werden.

❑ **ped_Close 0x0010 (16)** Ein Adreßzeiger auf eine spezifische Close-Routine des Druckertreibers, die bei der *CloseDevice*-Routine mit auf-

gerufen wird. Ihr wird ebenfalls ein Adreßzeiger auf die zum PrtBase-Device gehörende *IOStdReq*-Datenstruktur übergeben.

❑ **ped_PrinterClass 0x0014 (20)**

Folgende Konstanten definieren, um welche Art von Drukker es sich handelt (zu den Definitionen in C und Assembler gehört jeweils das Präfix *PPC_*):

C	Modula-2	Dez	Beschreibung
BWALPHA	bwAlpha	0	Ein Schwarzweißdrucker der nur alphanumerische Zeichen ausdrucken kann.
BWGFX	bwGfx	1	Ein Schwarzweißdrucker, der auch grafikfähig ist.
COLORALPHA	2		Ein Farbdrucker, der nur alphanumerische Zeichen darstellen kann.
COLORGFX	colorGfx	3	Ein Farbdrucker, der auch grafikfähig ist.

❑ **ped_ColorClass 0x0015 (21)**

Folgende Konstanten definieren, welche Farben der Drucker zu Papier bringen kann (zu den Definitionen in C und Assembler gehört jeweils das Präfix *PCC_*):

C	Modula-2	Dez	Beschreibung
BW	bw	1	Nur Schwarzweißdruck möglich.
YMC	ymc	2	Nur Gelb/Magentarot/Zyan-Druck möglich.
YMC_BW	ymcBw	3	Gelb/Magentarot/Zyan- oder Schwarzweiß-Druck möglich.
YMCB	ymcb	4	Nur Gelb/Magentarot/Zyan/Schwarz-Druck möglich.
ADDITIVE	additive	8	Das Komplement zu YMCB, also Blau/Grün/Rot/Weiß-Druck möglich.
WB	wb	9	Das Komplement zu BW, also nur »Weiß/Schwarz-Druck« möglich.
BGR	bgr	10	Nur Blau/Grün/Rot-Druck möglich.
BGR_WB	bgrWb	11	Blau/Grün/Rot- oder »Weiß/Schwarz-Druck« möglich.
BGRW	bgrw	12	Nur Blau/Grün/Rot/Weiß/-Druck möglich.

❑ **ped_MaxColumns 0x0016 (22)**

Hier steht die Anzahl der Zeichen, die in eine Druckerzeile passen. Normalerweise sind dies 80 Zeichen pro Zeile.

- **ped_NumCharSets 0x0017 (23)** Hier steht die Anzahl der verschiedenen Zeichensätze (Pica, Elite, Draft etc.), die der Drucker darstellen kann.

- **ped_NumRows 0x0018 (24)** Bei einem Matrixdrucker steht hier die Anzahl der Grafikzeilen, die pro Druckzeile ausgegeben werden können. Bei einem 8-Nadel-Drucker sind dies acht Grafikzeilen pro Druckzeile, aber Achtung: ein 24-Nadel-Drucker quetscht nicht unbedingt auch 24 Grafikzeilen in eine Druckzeile!

- **ped_MaxXDots 0x001A (26)** Dieser Wert legt fest, wie viele Punkte in einer Zeile nebeneinander passen. Ein Matrixdrucker mit 80 Zeichen/Zeile unterstützt normalerweise Grafikausdrücke mit 640 (einfache Dichte), 1280 (doppelte Dichte) und häufig sogar 2560 (vierfache Dichte) Punkten pro Zeile.

- **ped_MaxYDots 0x001E (30)** Dieser Wert legt fest, wie viele Punkte in einer Zeile übereinander passen. Bei Matrixdruckern wird dieser Wert im allgemeinen durch die Anzahl der Nadeln beschränkt.

- **ped_XDotsInch 0x0022 (34)** Durch diesen Wert wird die tatsächlich verwendete Druckdichte in x-Richtung festgelegt. Die Angabe erfolgt in (Druck-)Punkten pro Inch (2,54 cm).

- **ped_YDotsInch 0x0024 (36)** Durch diesen Wert wird die tatsächlich verwendete Druckdichte in y-Richtung festgelegt. Die Angabe erfolgt in (Druck-)Punkten pro Inch (2,54 cm).

- **ped_Commands 0x0026 (38)** Dies ist ein Adreßzeiger auf eine Umsetzungstabelle für die Amiga-Kommandosequenzen, die im Printer-Device definiert sind. Die jeweils 4 Byte langen Einträge enthalten der Reihe nach die Umsetzungen der Amiga-spezifischen Sequenzen in eine Sequenz, die der zugehörige Drucker versteht. Unterstützt der Drucker ein bestimmtes Format nicht (z.B. Kommando 21: Schattendruck ein), so muß an der entsprechenden Stelle in der Tabelle der Wert 0xFF (255) stehen.

- **ped_DoSpecial 0x002A (42)** Ein Adreßzeiger auf die *DoSpecial*-Routine, mit der eigene Steuersequenzen ausgegeben werden können.

- **ped_Render 0x002E (46)** Ein Adreßzeiger auf die *Render*-Routine, die eine Hardcopy-Funktion realisiert.

- **ped_TimeoutSecs 0x0032 (50)** Ist der Drucker im OFF-LINE Zustand, so definiert dieser Wert die Anzahl der Sekunden, die verstreichen dürfen, bis er wieder in den ON-LINE Zustand versetzt wird. Wird diese Zeit überschritten, so kommt es zu einem »Printer-Trouble-Requester«.

- **ped_8BitChars 0x0036 (54)** Bei übergroßen Fonts steht hier ein Adreßzeiger auf eine ins entsprechende Format konvertierte Zeichenkette.

- **ped_PrintMode 0x003A (58)** In diesem Feld steht ein Wert ungleich null, wenn ein Text gedruckt wurde, sonst null.
- **ped_ConvFunc 0x003E (62)** Ein Adreßzeiger auf die Routine, die die Zeichenkette in *ped_8BitChars* herstellt.

PrinterSegment

Offset Hex	Dez	C (devices/prtbase.h)	Modula-2 (PrtBase)
		struct PrinterSegment	PrinterSegment =
		{	RECORD
0000	0	ULONG ps_NextSegment;	nextSegment:BPTR;
0004	4	ULONG ps_runAlert;	runAlert:LONGCARD;
0008	8	UWORD ps_Version;	version:CARDINAL;
000A	10	UWORD ps_Revision;	revision:CARDINAL;
000C	12	struct PrinterExtendedData ps_PED;	ped:PrinterExtendedData;
004E	78	};	END;

Datenstrukturenrefernz »PrtBase«: *PrinterData.*

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur entspricht bis auf den *ps_NextSegment*-Adreßzeiger dem Header des Druckertreibers.

- **ps_NextSegment 0x0000 (0)** Ein BPTR auf das nächste Segment. Er wird beim Löschen des Druckertreibers aus dem Speicher durch die *UnloadSeg*-Routine benötigt.
- **ps_runAlert 0x0004 (4)** Dieser 4 Byte lange Wert enthält die Assembler-Befehle:

  MOVEQ #0, d0;
  RTS;

  Dadurch wird erreicht, daß bei dem Versuch, den Druckertreiber als Programm zu starten, kein Systemabsturz, sondern nur eine entsprechende Fehlermeldung entsteht.
- **ps_Version 0x0008 (8)** An dieser Stelle steht die Versionsnummer des Druckertreibers.
- **ps_Revision 0x000A (10)** An dieser Stelle steht die Revisionsnummer des Druckertreibers.
- **ps_PED 0x000C (12)** Eine eingebundene *PrinterExtendedData*-Datenstruktur, die die zu dem Treiber gehörenden Informationen über den angeschlossenen Drucker enthält.

8

RigidDiskBlock

Offset		C (devices/bootblock.h)	Modular-2 (HardBlocks)
Hex	Dez	struct RigidDiskBlock {	RigidDiskBlock = RECORD
0000	0	ULONG rdb_ID;	id: ARRAY [0..3] OF CHAR;
0004	4	ULONG rdb_SummedLongs;	summedLongs: LONGCARD;
0008	8	ULONG rdb_ChkSum;	chkSum: LONGINT;
000C	12	ULONG rdb_HostID;	hostID: LONGCARD;
		HR t (↵)	blockBytes: LONGCARD;
0010	16	ULONG rdb_Flags;	flags: RigidDiskFlagSet;
0014	20	ULONG rdb_BadBlockList;	badBlockList: LONGCARD;
0018	24	ULONG rdb_PartitionList;	partitionList: LONGCARD;
001C	28	ULONG rdb_FilesSysHeaderList;	filesSysHeaderList: LONGCARD;
0020	32	ULONG rdb_DriveInit;	driveInit: LONGCARD;
0024	36	struct rdb_Reserved1 6 * 4;	reserved1: ARRAY [0..3] OF LONG-CARD;
003C	60	ULONG rdb_Cylinders;	cylinders: LONGCARD;
0040	64	ULONG rdb_Sectors;	sectors: LONGCARD;
0044	68	ULONG rdb_Heads;	Heads: LONGCARD;
0048	72	ULONG rdb_Interleave;	interleave: LONGCARD;
004C	76	ULONG rdb_Park;	park: LONGCARD;
0050	80	ULONG rdb_Reserved2[3];	reserved2: ARRAY [0..2] OF LONG-CARD;
005C	92	ULONG rdb_WritePreComp;	writePreComp: LONGCARD;
0060	96	ULONG rdb_ReducedWrite;	reducedWrite: LONGCARD;
0064	100	ULONG rdb_StepRate;	stepRate: LONGCARD;
0068	104	ULONG rdb_Reserved3[5];	reserved3: ARRAY [0..4] OF LONG-CARD;
007C	124	ULONG rdb_RDBBlocksLo;	blocksLo: LONGCARD;
0080	128	ULONG rdb_RDBBlockHi;	blockHi: LONGCARD;
0084	132	ULONG rdb_LowCylinder;	lowCylinder: LONGCARD;
0088	136	ULONG rdb_HiCylinder;	hiCylinder: LONGCARD;
008C	140	ULONG rdb_CylBlocks;	cylBlocks: LONGCARD;
0090	144	ULONG rdb_AutoParkSeconds;	autoParkSeconds: LONGCARD;
0094	148	ULONG rdb_Reserved4[2];	reserved4: ARRAY [0..1] OF LONG-CARD;
009C	156	char rdb_DiskVendor[8];	diskVendor: ARRAY [0..7] OF CHAR;
00A4	164	char rdb_DiskProduct[16];	diskProduct: ARRAY [0..15] OF CHAR;
00B4	180	char rdb_DiskRevision[4];	diskRevision: ARRAY [0..3] OF CHAR;
00B8	184	char rdb_ControllerVendor[8];	controllerVendor: ARRAY [0..7] OF CHAR;

Offset		C (devices/bootblock.h)		Modula-2 (HardBlocks)
Hex	Dez			
00C0	192	char	rdb_ControllerProduct[16];	controllerProduct: ARRAY [0..15] OF CHAR;
00D0	208	char	rdb_ControllerRevision[4];	controllerRevision: ARRAY [0..3] OF CHAR;
00D4	212	ULONG	rdb_Reserved5[10];	Reserved5: ARRAY [0..9] OF LONGCARD;
00FC	252	}		END;

Datenstrukturenreferenz Keine.

Routinenreferenz Keine.

Beschreibung

Diese Datenstruktur gehört zu den neueren und ist noch nicht in allen Include-Files der einzelnen Compiler enthalten. Sie definiert einen Block, in dem alle wichtigen Informationen über die Festplatte enthalten sind. Ein solcher Block muß sich auf Festplatte innerhalb der 16 ersten Blocks befinden.

❑ **rdb_ID 0x0000 (0)** Eine vier Zeichen lange Kennung, die den Wert »RDSK« haben muß.

❑ **rdb_SummedLongs 0x0004 (4)** Die Anzahl der Bytes dieser Datenstruktur, über die die Checksumme gebildet wird.

❑ **rdb_ChkSum 0x0008 (8)** Die Checksumme des Blocks.

❑ **rdb_HostID 0x000C (12)** Die SCSI-Zielkennung des Blocks.

❑ **rdb_Flags 0x0010 (16)** Einige Flags, die zusätzliche Informationen zur Initialisierung der Festplatte beinhalten. Es sind folgende Werte definiert (dazu gehört jeweils das Präfix *RDBFF_*):

C	Dez	Beschreibung
LAST	1	Es gibt keine weitere Festplatte an diesem Controller.
LASTLUN	2	Ist nur bei SCSI-Festplatten von Bedeutung, aber von welcher?
LASTID	4	Ist nur bei SCSI-Festplatten von Bedeutung, aber von welcher?
NORESELECT	8	Das Festplattenlaufwerk braucht bei wiederholtem Zugriff nicht noch einmal angewählt zu werden.
DISKID	16	Die Plattenkennung (in den Feldern rdb_Diskxxx) ist gültig.
CTRLRID	32	Die Controllerkennung (in den Feldern rdb_Controllerxxx) ist gültig.

❑ **rdb_BadBlockList 0x0014 (20)** Die Nummer des ersten Blocks einer Liste von *BadBlock-Block*-Datestrukturen.

8

- **rdb_PartitionList 0x0018 (24)** Die Nummer des ersten Blocks einer Liste von *PartitionBlock*-Datestrukturen. Diese beschreiben jeweils eine Plattenpartition.
- **rdb_FilesSysHeaderList 0x001C (28)** Die Nummer des ersten Blocks einer Liste von *FileSysHeaderBlock*-Datestrukturen.
- **rdb_DriveInit 0x0020 (32)** Hier können Initialisierungsdaten für die Partition stehen.
- **rdb_Reserved 0x0024 (36)** In der vorliegenden Version immer auf 0xFFFFFFFF setzen.

Beachte: Die Nachfolgenden Felder beschreiben die Hardware-Eigenschaften der Festplatte.

- **rdb_Cylinders 0x003C (60)** Die Anzahl der Zylinder, über die die Festplatte verfügt.
- **rdb_Sectors 0x0040 (64)** Die Anzahl der Sektoren pro Zylinder.
- **rdb_Heads 0x0044 (68)** Die Anzahl der Schreib-Lese-Köpfe, über die das Festplattenlaufwerk verfügt.
- **rdb_Interleave 0x0048 (72)** Der Interleave-Faktor für das Festplattenlaufwerk.
- **rdb_Park 0x004C (76)** Die Nummer des Zylinders, der für das »Parken« der Festplatte benutzt wird.
- **rdb_Reserved2 0x0050 (80)** Bis jemand das Gegenteil beweist, befindet sich in diesem Feld die Telefonnummer meiner Oma!
- **rdb_WritePreComp 0x005C (92)** ???
- **rdb_ReducedWrite 0x0060 (96)** ???
- **rdb_StepRate 0x0064 (100)** Hier steht die Schrittrate für den Spurwechsel.
- **rdb_Reserved3 0x0068 (104)** Für zukünftige Erweiterungen, aber auch *rdb_Reserved2*.

Die nachfolgenden Felder beschreiben die logische Unterteilung der Platte

- **rdb_RDBBlocksLo 0x007C (124)** Die Nummer des ersten Zylinders, der für die Systeminformation (z.B. *BadBlockBlock*-Datenstrukturen) reserviert ist.
- **rdb_RDBBlockHi 0x0080 (128)** Die Nummer des letzten Zylinders, der für die Systeminformation (z.B. *BadBlockBlock*-Datenstrukturen) reserviert ist.
- **rdb_LowCylinder 0084 (132)** Die Nummer des ersten Zylinders, der für die Daten freisteht.
- **rdb_HiCylinder 0088 (136)** Die Nummer des letzten Zylinders, der für die Daten freisteht.
- **rdb_CylBlocks 0x008C (140)** Die Anzahl der nutzbaren Blocks pro Zylinder.
- **rdb_AutoParkSeconds 0x0090 (144)** Steht hier eine Null, so wird kein »Autopark« ausgeführt. Sonst gibt dieses Feld die Anzahl der Sekunden an, die vor dem »Autopark« abzuwarten sind.
- **rdb_Reserved4 0x0094 (148)** Noch ein bißchen Platz für die Zukunft.

- **rdb_DiskVendor 0x009C (156)** Hier kann eine Zeichenkette mit dem Herstellernamen der Platte stehen.
- **rdb_DiskProduct 0x00A4 (164)** Hier kann eine Zeichenkette mit dem Produktnamen der Platte stehen.
- **rdb_DiskRevision 0x00B4 (180)** Hier kann eine Versionskennung der Platte stehen.
- **rdb_ControllerVendor 0x00B8 (184)** Hier kann eine Zeichenkette mit dem Herstellernamen des Controllers stehen.
- **rdb_ControllerProduct 0x00C0 (192)** Hier kann eine Zeichenkette mit dem Produktnamen des Controllers stehen.
- **rdb_ControllerRevision 0x00D0 (208)** Hier kann eine Versionskennung des Controllers stehen.
- **rdb_Reserved5 0x00D4 (212)** Für zukünftige Erweiterungen reserviert.

A GLOSSAR

Adreßregister: Der Prozessor MC68000 hat sieben Adreßregister, über die die Adressierung der Speicherstellen (Auslesen und Beschreiben) abläuft. Diese Adreßregister heißen A0 bis A6.

Adreßzeiger: Ein Adreßzeiger ist ein 4 Byte langer Wert, dessen Inhalt eine Speicheradresse angibt, er »zeigt« also auf diese Speicherstelle; *NULL*(^).

Alert: Eine Meldung in der Form der bekannten Guru-Meditation ist ein System-Alert.

allokieren: (von allocate, engl: Zuweisen) Um einen Speicherbereich für eigene Zwecke benutzen zu können, muß dieser reserviert, »diesem Zweck zugewiesen« werden – deallozieren(^).

Anwender-Modus: Dies ist der Standardmodus des Prozessors MC68000. Im Gegensatz zum Supervisor-Modus(^) steht in diesem Modus nicht der gesamte Befehlssatz zur Verfügung (ausgeschlossen sind beispielsweise die Befehle, die das Statusregister direkt verändern).

Array: Ein Variablen-Feld (in einer beliebigen Programmiersprache), sprich eine durchnumerierte Aufzählung.

Audio-Device: Dieses Device(^) steuert die gesamte Tonausgabe des Amiga.

Basispunkt: Positionen innerhalb eines Windows werden im allgemeinen relativ zu einem Basispunkt angegeben (typischerweise die obere linke Ecke).

BPTR: Dies ist eine spezielle Art von Adreßzeigern, die nur von DOS verwendet wird. Ein BPTR, der auf ein Objekt zeigt, hat den Wert Adresse durch 4. Das Objekt muß also immer an einer durch 4 teilbaren Adresse liegen.

Bitmap: Eine Bitmap besteht aus mehreren, sich überlappenden Bitplanes(^). So werden jedem Grafikpunkt mehr als ein Bit Information zugeordnet, wodurch ihm auch mehr Farben zugeordnet werden können.

Bitplane: Ein Speicherbereich, dessen binärer Inhalt als Grafik interpretiert wird. Ein gesetztes Bit entspricht dabei einem gesetzten Punkt in einer Schwarzweißgrafik. Da die Grafikdaten in der Bitplane zeilenweise eingetragen sind, müssen zur korrekten Wiedergabe die Dimensionen (Breite und Höhe) der Grafik bekannt sein, damit das jeweilige Zeilenende erkannt wird.

Blitter: Dies ist einer der Co-Prozessoren des Amiga. Seine Aufgabe besteht im wesentlichen aus dem schnellen Kopieren von Speicherbereichen, wie es z.B. beim Verschieben eines Grafikausschnitts nötig ist.

Bob: (Abk.: Blitter Objekt) Ein Grafikobjekt, das der Blitter(^) durch direktes Kopieren (Verschieben) innerhalb der Bitplanes(^) animiert, also bewegt.

Border: (engl.: Rand) Grundsätzlich können alle Displayelemente(^) mit einer Umrandung ausgestattet werden.

Boolean-Gadget: Ein Gadget(^), das auf einmaliges Anklicken reagiert (wie z.B. das Close-Gadget eines Fensters).

Boolean-Toggle-Gadget: Ein Boolean-Gadget(^), mit dem Funktionen ein- und ausgeschaltet werden können. Viele Kopierprogramme verfügen über solche Gadgets zum Aktivieren eines jeweiligen Laufwerks.

Boolescher Ausdruck: Ein Boolescher Ausdruck bestimmt einen Zustand, der nur einen von zwei Zuständen annehmen kann (0/1, TRUE/FALSE).

Bootblock: Der Block der Diskette, der angibt, ob von ihr nach einem Reset gebootet werden kann. Er liegt auf Track 00 Spur 0.

BSTR: Diese besondere Art der Speicherung von Zeichenketten wird von Amiga-DOS verwendet. Ein BSTR ist ein BPTR(^) auf einen Speicherbereich, in dem im ersten Byte die Länge der Zeichenkette, und erst nachfolgend die Zeichenkette selbst steht.

Byte: Dies ist nicht nur eine 8 Bit lange Binärzahl, sondern in vielen Programmiersprachen auch der entsprechende Datentyp(^), dem Werte zwischen 0 und 255 oder manchmal auch zwischen –128 und 127 zugeordnet werden können.

Char: (Kurzform von Charakter) Ein Byte, dessen Wert ein einzelnes (Schrift-)Zeichen repräsentiert.

Chip-RAM: Der Speicherbereich, auf den die Co-Prozessoren zugreifen können. Bei älteren Amigas sind dies die unteren 512 Kbyte, bei neueren das untere Mbyte.

Checkmark: Das Grafikzeichen, das in Pull-down-Menüs zum Sichtbarmachen von eingeschalteten Funktionen verwendet wird (Standardsymbol ist ein Haken).

Clip: Ein Clip ist ein Speicherbereich, der einen Teil einer Grafik oder eines Textes enthält und mit dem etwas Besonderes passieren soll (Markieren, Verschieben, Verdoppeln etc.).

Clipboard-Device: Das Clipboard-Device dient zur einfachen Verwaltung und Zwischenspeicherung von Datenblöcken.

Colormap: Eine Farbtabelle, die die RGB-Anteile der Farben, die den einzelnen Farbregistern zugeordnet sind, bestimmt.

A

Console-Device: Dieses Device(^) steuert die Tastaturabfrage.

Container: Der Bereich in einem Proportional-Gadget(^), in dem sich der Knopf(^) befindet.

Copper: Dies ist einer der Co-Prozessoren des Amiga. Seine Hauptaufgabe besteht aus der Darstellung des Displayspeichers(^) auf dem Display(^).

Copperliste: Eine Befehlsliste für den Copper(^). Normalerweise nach der Strahlenposition, an der die Befehle ausgeführt werden sollen, geordnet.

Datenstruktur: Eine Datenstruktur ist ein Verbund von mehreren Parametern, wie Variablen und Flags(^), die für einen bestimmten Zweck (Windows, Gadgets(^), Diskettenoperationen etc.) die entsprechenden Einstellungen (Dimensionen, Farbwerte, Adreßzeiger(^) etc.) bereithalten.

Datentyp: Von Compilern werden mehrere Datentypen unterstützt. Typische Standard-Datentypen in C sind BOOL, LONG, SHORT und WORD.

Datenregister: Der Prozessor MC68000 hat acht Datenregister (D0 bis D7), die zum Datenaustausch, vor allem aber auch für arithmetische Operationen, zur Verfügung stehen.

Deallokieren: Ein Speicherbereich, der durch Allokieren(^) reserviert wurde, kann durch Deallokieren wieder freigegeben werden.

Default: (engl.: Versäumnis; sinngemäß: Voreingestellt) Ein Default-Parameter wird dann benutzt, wenn in einem entsprechenden Feld kein aktueller Parameter angegeben ist. Wird beispielsweise für ein neues Window kein Titeltext angegeben, so wird der Default-Titel aus der zugehörigen *Screen*-Datenstruktur verwendet.

Device: (engl.: Vorrichtung, aber auch Erfindung) Die Devices des Amiga sind in erster Linie Gerätetreiber, die die Ein- oder Ausgabe eines bestimmten Kanals (serielle und parallele Schnittstelle, Tonausgabe, Tastatureingabe etc.) steuern.

Diskfont-Library: Diese Library(^) verwaltet die verschiedenen Zeichensätze, die von Diskette geladen werden können.

Display: Eine Ausgabe auf dem Display ist nichts anderes als eine Ausgabe auf dem Bildschirm Ihres Monitors oder was auch immer Sie als Displaygerät verwenden.

Displayelement: Die Displayelemente des Amiga sind im wesentlichen Screens, Windows, Gadgets(^) und Requester(^).

Displayspeicher: (sinngemäß: Bildschirmspeicherbereich) Der Speicherbereich, dessen binärer Inhalt direkt als Grafik auf dem Display(^) ausgegeben wird.

Doppelwort: Langwort(^).

DOS-Library: Diese Library steuert die Ein- und Ausgabe über die Diskettenlaufwerke und Festplatten.

Event: Eine Art von Ereignisse, die auftreten können: Anklicken eines Gadgets; Drücken einer Taste; Systemfehler; etc.

Exception: Ein lokales, taskinternes Interrupt. Es kann durch ein Signal ausgelöst werden.

Exclude, EXCL: (engl.: Ausschließen) Modula-2-Befehl zum Entfernen eines Elementes aus einer Menge. Wird in den Datenstrukturen(^) verwendet, um aus einem *SET OF xxxFlags* ein oder mehrere Bits zu entfernen.

A

Exec-Library: Die Library(^), die für die interne Verwaltung, in erster Linie für die von Speicherbereichen anderer Libraries und Devices, zuständig ist.

Expansion-Library: Die Library(^), die für die Verwaltung der Erweiterungssteckkarten zuständig ist.

Farbregister: Wenn es bei der Ausgabe auf dem Display(^) um Farbzuordnungen geht, so werden grundsätzlich die Nummern von einem der 32 Farbregister verwendet. In diesen Speicherstellen steht dann einer der 4096 Farbwerte.

Farbwert: Beim Amiga eine 3 Byte lange Zahl, die eine der 4096 Farben definiert. Die Bytes definieren jeweils durch Zahlen zwischen 0 und 15 den Rot-, Grün- und Blauanteil der Farbe.

Fast-RAM: Der Speicherbereich, auf den die Sonderchips des Amiga keinen Zugriff haben.

FFP-Zahl: Eine 4 Byte lange Gleitkommazahl, die vom Amiga unterstützt wird und für die ein paar sehr schnelle Rechenroutinen zur Verfügung stehen. Der Nachteil liegt in der größeren Ungenauigkeit.

FIFO-Schlange: Ein First-In-/First-Out-Speicherbereich; das zuerst hineingeschriebene Element wird auch als erstes wieder ausgelesen.

Flag: (engl.: Flag) Eine 1-Byte-Marke enthält 8 Bit, diese werden als Flags angesehen, die einzeln gesetzt oder gelöscht werden können.

Font: (engl.: Zeichensatz) Auf jeder System-Diskette befinden sich in dem »fonts« Unterverzeichnis meist mehrere Zeichensätze, die vom Amiga anstelle des System-Fonts verwendet werden können.

Gadget: (engl.: Dingsbums; hier Symbolfeld) Gadgets sind Felder, die durch Anklicken mit der Maus eine bestimmte Aktion auslösen. Die meisten Fenster verfügen über Close- (oben links zum Schließen), Drag- (in der Titelleiste zum Verschieben), Depth- (oben rechts zum Schalten in den Vorder- oder Hintergrund) und Sizing-Gadgets (unten rechts zum Verändern der Größe).

Gameport-Device: Dieses Device(^) verwaltet die Eingaben der Gameports (auch als Joystickports oder Mouseports bekannt).

GEL: Graphics ELement. Eine Bezeichnung für einen beliebigen Bestandteil des Animationssystems.

GimmeZeroZero-Window: Bei dieser Art von Fenstern brauchen Sie sich bei Positionsangaben nicht um den Rand zu kümmern, da sich Angaben auf das innere Fenster (ohne den Rand) beziehen. Der Koordinatenursprung liegt dabei in der inneren linken oberen Ecke.

Graphics-Library: In dieser Library(^) sind die zur Ausgabe auf dem Display benötigten Routinen und Datenstrukturen definiert.

GZZ-Window: GimmeZeroZero-Window(^).

Handler: (engl.: Betreuer). Bezeichnung für ein Programm, das für die Handhabung eines Devices oder Resources zuständig ist.

Highlightning: (engl.: hervorheben) Beispiel: Ein durch Anklicken aktiviertes Gadget oder ein ausgewählter Menüpunkt wird oftmals durch Farbwechsel hervorgehoben.

HiRes: Der Grafikmodus, bei dem 640 Punkte (entspricht 80 Standardzeichen pro Zeile) horizontal dargestellt werden können.

Hot-Spot: Der Punkt im Mouse-Pointer, mit dem die Auswahl der Gadgets oder Menüpunkte stattfindet. Normalerweise liegt der Hot-Spot in der Spitze eines Pfeils oder in der Mitte eines Fadenkreuzes.

Icon: (engl.: »Heiligenbild«; sinngemäß: Piktogramm) Die zu einzelnen Disketten, Schubladen oder Programmen gehörenden Piktogramme, die von der Workbench aus durch doppeltes Anklicken aktiviert werden.

IDCMP: Abkürzung für »Intuition Direct Communications Message Port«. Dieser ermöglicht in erster Linie die Abfrage von Benutzertätigkeiten wie Mausbewegungen oder Tastatureingaben.

IEEE-Zahl: Ein Amiga-internes Gleitpunktzahlformat, für das einige Rechenroutinen zur Verfügung stehen. Da dieses Format länger als das der FFP-Zahlen(^) ist, ist die Rechengenauigkeit höher, jedoch dauert das Rechnen auch länger.

IFF: Abkürzung für »Interchange File Format«. Eine Norm zur einheitlichen Erstellung von Dateien mit Grafik, Text oder Sound.

Image: (engl.: Bild) »Kleinere« Grafiken werden allgemein als Image bezeichnet. Sie können durch Image-Datenstrukturen definiert sein.

Include, INCL: (engl.: einschließen) Modula-2-Befehl zum Hinzufügen eines Elementes in eine Menge. Wird in den Datenstrukturen(^) verwendet, um in einem *SET OF xxxFlags* ein oder mehrere Bits zu setzen.

Include-File: Die C- und Assembler-Compiler können auf diese Files, in denen Teile der einzelnen Libraries und Devices nochmals definiert sind, zugreifen.

Input-Device: Dieses Device(^) ist für die Eingabe zuständig.

Integer-Zahl: Eine ganzzahlige Zahl.

Interrupt: Eine Unterbrechung des Programms, die direkt am Prozessor ausgelöst wurde.

Intuition: Der Teil des Betriebssystems des Amiga, der für die Handhabung der Benutzeroberfläche mit all seinen Screens und Windows zuständig ist.

Intuition-Library: In dieser Library(^) sind die Routinen und Datenstrukturen der Intuition(^) definiert.

KeyMap: Eine Konvertierungsliste, die den Tasten und Tastenkombinationen ein oder mehrere Zeichen (Chars(^)) zuordnet.

Keyboard-Device: Dieses Device(^) ist für die Eingabe von Zeichen über die Tastatur zuständig.

Knopf: Der Teil eines Proportional-Gadgets(^), der verschoben werden kann, um dadurch einen Wert einzustellen.

Langwort: Ein 32 Bit (4 Byte) langer Wert.

Layer: (engl.: Schicht) Ein Displayelement des grafischen Systems, das zu einem beliebigen Rechteck gehört.

Library: (engl.: Bibliothek) Die Routinen, die uns das Betriebssystem zur Verfügung stellt, sind nach Funktionsgruppen geordnet und dementsprechend in Libraries unterteilt.

Lock: Ein Lock ist eine Datenstruktur aus der DOS-Library, die den Zugriff mehrerer Multitasking-Komponenten auf eine Datei koordiniert. Sie dient auch als Identifikation dieser Datei.

Longword alligned: Das zugehörige Objekt liegt an einer durch 4 teilbaren Adresse im Speicher.

LoRes: Der Grafikmodus, bei dem 320 Punkte (60 Standardzeichen pro Zeile) horizontal dargestellt werden können.

Marke: Maske (^).

Maske: Eine Maske gibt die relevanten Bits eines Ausdrucks an.

Message: Eine Nachricht in Form einer Message-Datenstruktur plus Daten, die ein Task(^) an einen anderen schickt.

Messageport: Die »Empfangsstation« eines Tasks(^), an die Nachrichten in Form von Messages(^) geschickt werden.

Mouse-Pointer: (engl.: Mauszeiger) Das Grafiksymbol, das Sie durch Bewegen der Maus auf dem Display(^) steuern.

Multitasking: Der Amiga verfügt über die Fähigkeit des Multitaskings, also mehrere Tasks(^) (Aufgaben, Programme) gleichzeitig bearbeiten zu können.

Narrator-Device: Dieses Device(^) ist für die Steuerung der Sprachausgabe zuständig.

NIL: (Abk.: Not in List) Modula-2-Format von *NULL*(^).

Nibble: Ein halbes Byte, also 4 Bit, nennt man auch ein Nibble. Ein Byte läßt sich somit in ein oberes und unteres Nibble unterteilen.

Node: (engl.: Knoten) Dient zur Verkettung einer Liste. Damit wird oft auch die Node-Datenstruktur (die dem gleichen Zweck dient) gemeint.

NULL: Im C-Format ist dies ein Adreßzeiger(^), der auf keinen Speicherplatz zeigt, also dessen Inhalt Null ist. (NIL(^))

Packets: So werden Nachrichten bzw. Befehle genannt, die in Form von *Packet*-Datenstrukturen zwischen Prozessen(^) ausgetauscht werden.

Parallel-Device: Dieses Device(^) ist für die Steuerung der parallelen Schnittstelle zuständig.

Pointer: Adreßzeiger(^).

Port: (engl.: Hafen) allgemein eine Schnittstelle.

Printer-Device: Dieses Device(^) ist für die Ansteuerung des Druckers zuständig.

Proportional-Gadget: Bei dieser Art von Gadgets(^) kann der Benutzer einen Wert durch Verschieben eines Knopfes, ähnlich wie bei einem Schiebepotentiometer, einstellen.

Prozeß: Ein Prozeß ist ein selbständiges, innerhalb des Multitasking ablaufendes Programm.

Rastport: Ein Rastport definiert zu einem Displayelement(^) den Zeichenstift, mit dem grafische Ausgaben getätigt werden.

Raw-Code: Die intern verwendete Codierung aller Tasten der Tastatur.

Refresh: (engl.: wiederauffrischen) Wird ein Teil eines Windows verdeckt (z.B. durch Verkleinern oder durch Überlappen eines anderen Windows) und anschließend wieder freigelegt, so ist ein Refresh notwending, um den alten Inhalt des Windows wiederherzustellen.

Resources-Library: Diese Library ermöglicht die Handhabung neuer Geräte.

A

Requester: (engl.: Aufforderer; sinngemäß: Kommunikationsfenster) Ein Requester ist ein besonderes Aufforderungsfenster, das den Benutzer auf wichtige Mißstände hinweist (z.B. »Volume dfx: is full«).

Segment: Falls ein Programm (oder Daten) nicht in einen zusammenhängenden Speicherbereich paßt, wird es in mehrere verkettete Bereiche (Segmente) geladen.

Serial-Device: Dieses Device(^) ist für die Steuerung der seriellen Schnittstelle zuständig.

String: Zeichenkette(^).

String-Gadget: Ein Gadget(^), mit dem der Benutzer eine Zeichenkette eingeben oder eine bereits bestehende verändern kann. BEISPIEL: Das Gadget, das bei der Auswahl des Menüpunktes »Rename« auf der Workbench erscheint.

Supervisor-Modus: Der Prozessor MC68000 hat mehrere Arbeitsmodi. In diesem Überwachungsmodus können Sie im Gegensatz zu dem Anwender-Modus(^) über alle Befehle des Prozessors verfügen.

System-Gadget: Die Gadgets, die bereits vom System definiert sind und nur noch über Flags(^) eingeschaltet werden müssen. BEISPIEL: Das Sizing-Gadget, das sich in der unteren rechten Ecke von Fenstern befindet.

Tastaturpuffer: Der Speicherbereich, der alle Tastatureingaben zwischenspeichert, bis sie abgerufen werden.

Task: (engl.: Aufgabe) Der Amiga ist bekanntermaßen Multitaskingfähig, das heißt, er kann mehrere Aufgaben gleichzeitig bearbeiten. Ein Task ist also nichts weiteres als eine Aufgabe (Programm), die parallel zu anderen ausgeführt werden kann (dabei wird in Wirklichkeit nicht parallel gearbeitet, sondern lediglich zwischen den einzelnen Tasks hin- und hergeschaltet).

Terminator: Abbruchzeichen; Wird bei der Datenübertragung über die serielle oder parallele Schnittstelle ein Terminator gesendet, so wird dieser als das letzte Zeichen erkannt.

Timer-Device: Mit diesem Device(^) können beliebige Zeiteinheiten mit Hilfe der Timer gemessen werden.

Trackdisk-Device: Mit diesem Device(^) können die Diskettenlaufwerke angesteuert werden.

Trap: Eine durch einen unzulässigen Prozessorbefehl ausgelöste Programmunterbrechung.

Überwachungs-Modus: Supervisor-Modus(^).

Underscore: Dieses Zeichen »_« ist ein Underscore.

Unit: Ein Device kann mehrmals göffnet werden. Man sagt dann, daß mehrere Units existieren.

View: Ein View übernimmt die Verwaltung des Gesamtdisplays. Auf ihm können mehrere Viewports(^) geöffnet werden.

Viewport: Ein Viewport ist ein Displayelement(^) des grafischen Systems und stellt die Verbindung zwischen dem Displayspeicher(^), den Bitmaps(^) und Colormaps(^) dar.

Volume: (engl: Lautstärke, tut aber nichts zur Sache! Hier ist wohl eher ein Band gemeint). So wird im Amiga-DOS jeder eigenständiger Datenträger z.B. Diskette, Festplatte, aber auch das Laufwerk selbst bezeichnet.

Wort (Word): Ein 16 Bit (2-Byte) langer Wert.

Wordalligned: Das zugehörige Objekt liegt an einer durch 2 teilbaren Adresse im Speicher.

Workbench-Library: In dieser Library(^) sind alle die Funktionen und Datenstrukturen definiert, die zur Verwaltung der Workbench nötig sind.

Zeichenkette, Null-terminierte: Eine Zeichenkette, bestehend aus mehreren Chars, deren letzter Eintrag null ist.

B LITERATURVERZEICHNIS

(1) [KRA] Rom Kernal Reference Manual includes & Autodocs
(Verlag Addison Wesley)

(2) [KRE] Rom Kernal Reference Manual Exec
(Verlag Addison Wesley)

(3) [KRL] Rom Kernal Reference Manual Libraries & Devices
(Verlag Addison Wesley)

(4) [HRM] Hardware Reference Manual
(Verlag Addison Wesley)

(5) [IRM] Intuition Reference Manual
(Verlag Addison Wesley)

(6) [ADH] Amiga-DOS-Handbuch
(Verlag Markt & Technik)

(7) [APH] Amiga-Programmier-Handbuch
(Verlag Markt & Technik)

(8) [APM] Amiga-Programmieren in Maschinensprache
(Verlag Markt & Technik)

(9) [APH] Amiga-Programmierpraxis-Intuition
(Verlag Markt & Technik)

(10) [ABP] Amiga-Basic-Profibuch

(Verlag Maxon)

(11) [AI1] Amiga Intern Band 1

(Verlag Data Becker)

(12) [AI2] Amiga Intern Band 2

(Verlag Data Becker)

B

C ABBILDUNGSVERZEICHNIS

D QUICK-REFERENCE

Dieser Anhang dient der Schnellübersicht über die einzelnen Datenstrukturen, die in diesem Buch behandelt werden. In der ersten Zeile finden Sie den »C«-Namen der Datenstruktur und das Include-File, in dem diese Datenstruktur definiert ist. Darunter steht der Modula-2-Name und der Name des Definitions-Moduls, aus dem diese Datenstruktur importiert werden kann. Es folgt eine Auflistung der Namen der einzelnen Komponenten. Wie gewohnt, in der Reihenfolge hexadezimaler Offset, »C«- und Modula-2-Format.

AnimComp (graphics/gels.h)
AnimComp (Graphics) 134

```
0000 Flags flags
0002 Timer timer
0004 TimeSet timeSet
0006 NextComp nextComp
000A PrevComp prevComp
000E NextSeq nextSeq
0012 PrevSeq prevSeq
0016 AnimCRoutine animCRoutine
001A YTrans yTrans
001C XTrans xTrans
001E HeadOb headOb
0022 AnimBob animBob
0026
```

AnimOb (graphics/gels.h)
AnimOb (Graphics) 136

```
0000 NextOb nextOb
0004 PrevOb prevOb
0008 Clock clock
000C AnOldY anOldY
000E AnOldX anOldX
0010 AnY anY
0012 AnX anX
0014 YVel yVel
0016 XVel xVel
0018 YAccel yAccel
001A XAccel xAccel
001C RingYTrans ringYTrans
001E RingXTrans ringXTrans
0020 AnimORoutine animORoutine
0024 HeadComp headComp
0028 AUserExt userExt
002A
```

AreaInfo (graphics/rastport.h)
AreaInfo (Graphics) 139

```
0000 VctrTbl vctrTbl
0004 VctrPtr vctrPtr
0008 FlagTbl flagTbl
000C FlagPtr flagPtr
0010 Count count
0012 MaxCount maxCount
0014 FirstX firstX
0016 FirstY firstY
0018
```

AvailFonts (libraries/diskfont.h)
AvailFont (DiskFont) 223

```
0000 af_Type type
0002 af_Attr attr
000A
```

AvailFontsHeader (libraries/diskfont.h)
AvailFontHeader (DiskFont) 224

```
0000 afh_NumEntries numEntries
0002
```

BadBlockBlock (devices/hardblocks.h)
BadBlockBlock (Hardblocks) 323

```
0000 bbb_ID id
0004 bbb_SummedLongs summedLongs
0008 bbb_ChkSum chkSum
000C bbb_HostID hostID
0010 bbb_Next next
0014 bbb_Reserved reserved
0018 bbb_BadBlockPairs blockPairs
0200
```

BadBlockEntry (devices/hardblocks.h)
BadBlockEntry (Hardblocks) 325

```
0000 bbe_BadBlock badBlock
0004 bbe_GoodBlock goodBlock
0008
```

BitMap (graphics/gfx.h)
BitMap (Graphics) 140

```
0000 BytesPerRow bytesPerRow
0002 Rows rows
0004 Flags flags
0005 Depth depth
0006 pad pad
0008 Planes planes
0028
```

bltnode (hardware/blit.h)
Bltnode (Hardware) 235

```
0000 n n
0004 function function
0008 stat stat
000A blitsize blitsize
000C beamsync beamsync
000E cleanup cleanup
0012
```

Bob (graphics/gels.h)
Bob (Graphics) 141

```
0000 Flags flags
0002 SaveBuffer saveBuffer
0006 ImageShadow imageShadow
000A Before before
000C After after
0012 BobVSprite bobVSprite
0016 BobComp bobComp
001A DBuffer dBuffer
001E BUserExt
0020
```

BoolInfo (intuition/intuition.h)
BoolInfo (Intuition) 69

```
0000 Flags flags
0002 Mask mask
0006 Reserved reserved
000A
```

BootBlock (devices/bootblock.h)
BootBlock (BootBlock) 325

```
0000 bb_id id
0004 bb_chksum chkSum
0008 bb_dosblock dosBlock
000C
```

Border (intuition/intuition.h)
Border (Intuition) 70

```
0000 LeftEdge leftEdge
0002 TopEdge topEdge
0004 FrontPen frontPen
0005 BackPen backPen
0006 DrawMode drawMode
0007 Count count
0008 XY xy
000C NextBorder nextBorder
0010
```

CIA (hardware/cia.h)
CIAB (Hardware) 236

```
0000 ciapra pra
0001 pad0 pad0
0100 ciaprb prb
0101 pad1 pad1
0200 ciaddra ddra
0201 pad2 pad2
0300 ciaddrb ddrb
0301 pad3 pad3
0400 ciatalo talo
0401 pad4 pad4
0500 ciatahi tahi
0501 pad5 pad5
0600 ciatblo tblo
0601 pad6 pad6
0700 ciatbhi tbhi
0701 pad7 pad7
0800 ciatodlow todlow
0801 pad8 pad8
0900 ciatodmid todmid
0901 pad9 pad9
0A00 ciatodhi todhi
0A01 pad10 pad10
0B00 unusedreg unusedreg
0B01 pad11 pad11
0C00 ciasdr sdr
0C01 pad12 pad12
0D00 ciaicr icr
```

```
0D01 pad13 pad13
0E00 ciacra cra
0E01 pad14 pad14
0F00 ciacrb crb
0F02
```

ClipboardUnitPartial (devices/clipboard.h)
ClipboardUnitPartial (Clipboard) 283

```
0000 Node cu_Node node
000E cu_UnitNum unitNum
0012
```

ClipRect (graphics/clip.h)
ClipRect (Graphics) 144

```
0000 Next next
0004 prev prev
0008 lobs lobs
000C BitMap bitMap
0010 bounds bounds
0018 _p1 p1
001C _p2 p2
0020 reserved reserved
0024
```

CollTable (graphics/gels.h)
CollTable (Graphics) 145

```
0000 collPtrs collPtrs
0040
```

ColorMap (graphics/view.h)
ColorMap (Graphics) 146

```
0000 Flags flags
0001 Type type
0002 Count count
0004 ColorTable colorTable
0008
```

CommandLineInterface (libraries/dosextens.h)
CommandLineInterface (Dos) 197

```
0000 cli_Result2 result2
0004 cli_SetName setName
0008 cli_CommandDir commandDir
000C cli_ReturnCode returnCode
0010 cli_CommandName commandName
0014 cli_FailLevel failLevel
0018 cli_Prompt prompt
001B cli_StandardInput
     standardInput
0020 cli_CurrentInput current
     Input
0024 cli_CommandFile command
     File
0028 cli_Interactive inter-
     active
002B cli_Background back-
     ground
0030 cli_CurrentOutput
     currentOutput
0034 cli_DefaultStack
     defaultStack
0038 cli_StandardOutput
     standardOutput
003C cli_Module module
0040
```

ConfigDev (libraries/configvars.h)
ConfigDev (Expansion) 227

```
0000 cd_Node node
000E cd_Flags flags
000F cd_Pad pad
0010 cd_Rom rom
0020 cd_BoardAddr boardAddr
0024 cd_BoardSize boardSize
0028 cd_SlotAddr slotAddr
002A cd_SlotSize slotSize
002C cd_Driver driver
0030 cd_NextCD nextCD
0034 cd_Unused unused
```

ConUnit (devices/conunit.h)
ConUnit (ConUnit) 326

```
0000 cu_MP mp
0022 cu_Window window
0026 cu_XCP xCP
0028 cu_YCP yCP
002A cu_XMax xMax
002C cu_YMax yMax
002E cu_XRSize xRSize
0030 cu_YRSize yRSize
0032 cu_XROrigin xROrigin
0034 cu_YROrigin yROrigin
0036 cu_XRExtant xRExtant
0038 cu_YRExtant yRExtant
003A cu_XMinShrink xMinShrink
```

```
003C cu_YMinShrink yMinShrink
003E cu_XCCP xcCP
0040 cu_YCCP ycCP
0042 cu_KeyMapStruct keyMap
     Struct
0062 cu_TabStops tabStops
0102 cu_Mask mask
0103 cu_FgPen fgPen
0104 cu_BgPen bgPen
0105 cu_AOLPen aolPen
0106 cu_DrawMode drawMode
0107 cu_AreaPtSz areaPtSz
0108 cu_AreaPtrn areaPtrn
010C cu_Minterms minTerms
0114 cu_Font font
0118 cu_AlgoStyle algoStyle
0119 cu_TxFlags txFlags
011A cu_TxHeight txHeight
011C cu_TxWidth txWidth
011E cu_TxBaseline txBaseLine
0120 cu_TxSpacing txSpacing
0122 cu_Modes modes
0125 cu_RawEvents rawEvents
0128
```

copinit (graphics/copper.h)
Copinit (Graphics) 147

```
0000 diagstrt diagstrt
0008 sprstrtup sprstrtup
0058 sprstop sprstop
005C
```

CopIns (graphics/copper.h)
CopIns (Graphics) 147

```
0000 OpCode opCode
0002 nxtlist nxtlist
0002 VWaitPos vWaitPos
0002 DestAddr destAddr
0004 HWaitPos hWaitPos
0004 DestData destData
0006
```

CopList (graphics/copper.h)
CopList (Graphics) 149

```
0000 Next next
0004 _CopList copList
0008 _ViewPort viewPort
000C CopIns copIns
0010 CopPtr copPtr
0014 CopLStart copLStart
0018 CopSStart copSStart
001C Count count
001E MaxCount maxCount
0020 DyOffset dyOffset
0022
```

cprlist (graphics/copper.h)
Cprlist (Graphics) 150

```
0000 Next next
0004 start start
0008 MaxCount maxCount
000A
```

CurrentBinding (libraries/configvars.h)
CurrentBinding (Expansion) 228

```
0000 cb_ConfigDev configDev
0004 cb_FileName fileName
0008 cb_ProductString
     productString
000C cb_ToolTypes toolTypes
0010
```

Custom (hardware/custom.h)
Custom (Hardware) 243

```
0000 bltddat bltddat
0002 dmaconr dmaconr
0004 vposr vposr
0006 vhposr
0008 dskdatr dskdatr
000A joy0dat joy0dat
000C joy1dat joy1dat
000E clxdat clxdat
0010 adkconr adkconr
0012 pot0dat pot0dat
0014 pot1dat pot1dat
0016 potinp potinp
0018 serdatr serdatr
001A dskbytr dskbytr
001C intenar intenar
001E intreqr intreqr
0020 dskpt dskpt
0024 dsklen dsklen
0026 dskdat dskdat
0028 refptr refptr
002A vposw vposw
002C vhposw vhposw
```

D

```
002E copcon copcon
0030 serdat serdat
0032 serper serper
0034 potgo potgo
0036 joytest joytest
0038 strequ strequ
003A strvbl strvbl
003C strhor strhor
003E strlong strlong
0040 bltcon0 bltcon0
0042 bltcon1 bltcon1
0044 bltafwm bltafwm
0046 bltalwm bltalwm
0048 bltcpt bltcpt
004B bltbpt bltbpt
0050 bltapt bltapt
0054 bltdpt bltdpt
0058 bltsize bltsize
005A pad2d pad2d
0060 bltcmod bltcmod
0062 bltbmod bltbmod
0064 bltamod bltamod
0066 bltdmod bltdmod
0068 pad34 pad34
0070 bltcdat bltcdat
0072 bltbdat bltbdat
0074 bltadat bltadat
0076 pad3b pad3b
007E dsksync dsksync
0080 cop1lc cop1lc
0084 cop2lc cop2lc
0088 copjmp1 copjmp1
008A copjmp2 copjmp2
008C copins copins
008E diwstrt diwstrt
0090 diwstop diwstop
0092 ddfstrt ddfstrt
0094 ddfstop ddfstop
0096 dmacon dmacon
0098 clxcon clxcon
009A intena intena
009C intreq intreq
009E adkcon adkcon
00A0 aud aud
00E0 bplpt bplpt
00F8 pad7c pad7c
0100 bplcon0 bplcon0
0102 bplcon1 bplcon1
0104 bplcon2 bplcon2
0106 pad83 pad83
0108 bpl1mod bpl1mod
010A bpl2mod bpl2mod
010C pad86 pad86
0110 bpldat bpldat
011C pad8e pad8e
0120 sprpt sprpt
0180 color color
01C0
```

DateStamp (libraries/dos.h)
Date (Dos) 199

```
0000 ds_Days days
0004 ds_Minute minute
0008 ds_Tick tick
000C
```

DBufPacket (graphics/gels.h)
DBufPacket (Graphics) 151

```
0000 BufY bufY
0002 BufX bufX
0004 BufPath bufPath
0008 BufBuffer bufBuffer
000C
```

Device (exec/devices.h)
Device (Exec) 26

```
0000 dd_Library library
0022
```

DeviceData (devices/prtbase.h)
DeviceData (PrtBase) 339

```
0000 dd_Device device
0022 dd_Segment segment
0026 dd_ExecBase execBase
002A dd_CmdVectors cmdVectors
002E dd_CmdBytes cmdBytes
0032 dd_NumCommands numCommands
0034
```

DeviceList (libraries/dos.h)
DeviceList (Dos) 200

```
0000 dl_Next next
0004 dl_Type type
0008 dl_Task task
000C dl_Lock lock
```

```
0010 dl_VolumeDate volumeDate
001C dl_LockList lockList
0020 dl_DiskType diskType
0024 dl_unused unused
0028 dl_Name name
002C
```

DeviceNode (libraries/dos.h)
DeviceList (Dos) 202

```
0000 dn_Next next
0004 dn_Type type
0008 dl_Task task
000C dn_Lock lock
0010 dn_Handler handler
0014 dn_StackSize stackSize
0018 dn_Priority priority
001C dn_Startup startup
0020 dn_SegList segList
0024 dn_GlobVec globVec
0028 dn_Name name
002C
```

DiagArea (libraries/configregs.h)
DiagArea (Expansion) 229

```
0000 da_Config config
0001 da_Flags flags
0002 da_Size size
0004 da_DiagPoint diagPoint
0006 da_BootPoint bootPoint
0008 da_Name name
000A da_Reserved01 reserved01
000C da_Reserved02 reserved02
000E
```

DiscResource (resources/disc.h)
DiscResource (Resources) 268

```
0000 dr_Library library
0022 dr_Current current
0026 dr_Flags flags
0027 dr_pad pad
0028 dr_SysLib sysLib
002C dr_CiaResource ciaResource
0030 dr_UnitID unitID
0040 dr_Waiting waiting
004E dr_DiscBlock discBlock
0064 dr_DiscSync discSync
007A dr_Index index
0090
```

DiscResourceUnit (resources/disc.h)
DiscResourceUnit (Resources) 269

```
0000 dru_Message message
0014 dru_DiscBlock discBlock
002A dru_DiscSync discSync
0040 dru_Index index
0056
```

DiskFontHeader (libraries/diskfont.h)
DiskFontHeader (DiskFont) 225

```
0000 dfh_DF df
000E dfh_FileID fileId
0010 dfh_Revision revision
0012 dfh_Segment segment
0016 dfh_Name name
0036 dfh_TFtf
006A
```

DiskObject (workbench/workbench.h)
DiskObject (Workbench) 271

```
0000 do_Magic magic
0002 do_Version version
0004 do_Gadget gadget
0030 do_Type type
0032 do_DefaultTool defaultTool
0036 do_ToolTypes toolTypes
003A do_CurrentX currentX
003E do_CurrentY currentY
0042 do_DrawerData drawerData
0046 do_ToolWindow toolWindow
004A do_StackSize stackSize
004E
```

DosEnvec (libraries/filehandler.h)
DosEnvec (DOS) 204

```
0000 de_TableSize
0004 de_SizeBlock
0008 de_SecOrg
000C de_Surfaces
0010 de_SectorPerBlock
0014 de_BlocksPerTrack
0018 de_Reserved
001C de_PreAlloc
0020 de_Interleave
0024 de_LowCyl
0028 de_HighCyl
002C de_NumBuffers
0030 de_BuffMemType
0034 de_MaxTransfer
```

0038 de_Mask
003C de_BootPri
0040 de_DosType
0044

DosInfo (libraries/dosextens.h)
DosInfo (Dos) 206

0000 di_McName mcName
0004 di_DevInfo devInfo
0008 di_Devices devices
000C di_Handlers handlers
0010 di_NetHand netHand
0014

DosLibrary (libraries/dosextens.h)
DosLibrary (Dos) 206

0000 dl_lib lib
0022 dl_Root root
0026 dl_GV gv
002A dl_A2 a2
002E dl_A5 a5
0032 dl_A6 a6
0036

DosPacket (libraries/dosextens.h)
DosPacket (Dos) 207

0000 dp_Link link
0004 dp_Port port
0008 dp_Type type
000C dp_Res1 res1
0010 dp_Res2 res2
0014 dp_Arg1 arg1
0018 dp_Arg2 arg2
001C dp_Arg3 arg3
0020 dp_Arg4 arg4
0024 dp_Arg5 arg5
0028 dp_Arg6 arg6
002C dp_Arg7 arg7
0030

DrawerData (workbench/workbench.h)
DrawerData (Workbench) 274

0000 dd_NewWindow newWindow
0030 dd_CurrentX currentX
0034 dd_CurrentY currentY
0038

ExecBase (exec/execbase.h)
ExecBase (Exec) 26

0000 LibNode libNode
0022 SoftVer softVer
0024 LowMemChkSum lowMemChkSum
0026 ChkBase chkBase
002A ColdCapture coldCapture
002E CoolCapture coolCapture
0032 WarmCapture warmCapture
0036 SysStkUpper sysStkUpper
003A SysStkLower sysStkLower
003E MaxLocMem maxLocMem
0042 DebugEntry debugEntry
0046 DebugData debugData
004A AlertData alertData
004E MaxExtMem maxExtMem
0052 ChkSum chkSum
0054 IntVects intVects
0114 ThisTask thisTask
0118 IdleCount idleCount
011C DispCount dispCount
0120 Quantum quantum
0122 Elapsed elapsed
0124 SysFlags sysFlags
0126 IDNestCnt idNestCnt
0127 TDNestCnt tdNestCnt
0128 AttnFlags attnFlags
012A AttnResched attnResched
012C ResModules resModules
0130 TaskTrapCode taskTrapCode
0134 TaskExceptCode
taskExceptCode
0138 TaskExitCode taskExitCode
013C TaskSigAlloc taskSigAlloc
0140 TaskTrapAlloc
taskTrapAlloc
0142 MemList memList
0150 ResourceList resourceList
015E DeviceList deviceList
016C IntrList intrList
017A LibList libList
0188 PortList portList
0196 TaskReady taskReady
01A4 TaskWait taskWait
01B2 SoftInts softInts
0202 LastAlert lastAlert
0212 VBlankFrequency
vBlankFrequency
0213 PowerSupplyFrequency
powerSupplyFrequency
0214 SemaphoreList
semaphoreList

```
0222 KickMemPtr kickMemPtr
0226 KickTagPtr kickTagPtr
022A KickCheckSum kickCheckSum
022C ExecBaseReserved
     execBaseReserved
0238 ExecBaseNewReserved
     execBaseNewReserved
024C
```

ExpansionBase (libraries/expansionbase.h) 230

```
0000 LibNode
0022 Flags
0023 pad
0024 ExecBase
0028 SegList
002C CurrentBinding
003C BoardList
004A MountList
0058 AllocRTable
0158 BindSemaphore
0186 Int2List
019C Int6List
01B2 Int7List
01C8
```

ExpansionControl (libraries/configregs.h)
ExpansionControl (Expansion) 232

```
0000 ec_Interrupt interrupt
0001 ec_Reserved11 reserved11
0002 ec_BaseAddress baseAddress
0003 ec_Shutup shutup
0004 ec_Reserved14 reserved14
0005 ec_Reserved15 reserved15
0006 ec_Reserved16 reserved16
0007 ec_Reserved17 reserved17
0008 ec_Reserved18 reserved18
0009 ec_Reserved19 reserved19
000A ec_Reserved1a reserved1a
000B ec_Reserved1b reserved1b
000C ec_Reserved1c reserved1c
000D ec_Reserved1d reserved1d
000E ec_Reserved1e reserved1e
000F ec_Reserved1f reserved1f
0010
```

ExpansionRom (libraries/configregs.h)
ExpansionRom (Expansion) 233

```
0000 er_Type type
0001 er_Product product
0002 er_Flags flags
0003 er_Reserved03 reserved03
0004 er_Manufacturer
     manufacturer
0006 er_SerialNumber
     serialNumber
000A er_InitDiagVec
     initDiagVec
000C er_Reserved0c reserved0c
000D er_Reserved0d reserved0d
000E er_Reserved0e reserved0e
000F er_Reserved0f reserved0f
0010
```

FatIntuiMessage (intuition/intuition.h)
FatIntuiMessage (Intuition) 71

```
0000 IntuiMessage intuiMessage
0034 PrevKeys prevKeys
0038
```

FileHandle (libraries/dosextens.h)
FileHandle (Dos) 209

```
0000 fh_Link link
0004 fh_Port port
0008 fh_Type type
000C fh_Buf buf
0010 fh_Pos pos
0014 fh_End end
0018 fh_Funcs func1
001C fh_Func2 func2
0020 fh_Func3 func3
0024 fh_Args arg1
0028 fh_Arg2 arg2
002C
```

FileInfoBlock (libraries/dos.h)
FileInfoBlock (Dos) 210

```
0000 fib_DiskKey diskKey
0004 fib_DirEntryType
     dirEntryType
0008 fib_FileName fileName
0074 fib_Protection protection
0078 fib_EntryType entryType
007C fib_Size size
0080 fib_NumBlocks numBlocks
0084 fib_Date date
0090 fib_Comment comment
00E0 fib_Reserved padding
0104
```

D

FileLock (libraries/dosextens.h)
FileLock (Dos) 211

```
0000 fl_Link link
0004 fl_Key key
0008 fl_Access access
000C fl_Task task
0010 fl_Volume volume
0014
```

FileSysHeaderBlock (devices/hardblocks.h)
FileSysHeaderBlock (Hardblocks) 329

```
0000 fhb_ID id
0004 fhb_SummedLongs summedLongs
0008 fhb_ChkSum chkSum
000C fhb_HostID hostID
0010 fhb_Next next
0014 fhb_Flags flags
0018 fhb_Reserved reserved 1
0020 fhb_DosType dosType
0024 fhb Version version
0028 fhb_PatchFlags patchFlags
002C fhb_Type type
0030 fhb_Task task
0034 fhb_Lock lock
0038 fhb_Handler handler
003C fhb_StackSize stackSize
0040 fhb_Priority priority
0044 fhb_Startup startup
0048 fhb_SegListBlocks seg-
     ListBlocks
004C fhb_GlobVec globVec
0060 fhb_Reserved2 reserved2
00AC fhb_Reserved3 reserved3
0100
```

FileSystemStartupMsg (libraries/filehandler.h)
FileSystemStartupMsg (DOS) 212

```
0000 fssm_Unit unit
0004 fssm_Device device
0008 fssm_Environ environ
000C fssm_Flags flags
0010
```

FontContents (libraries/diskfont.h)
FontContents (DiskFont) 225

```
0000 fc_FileName fileName
0100 fc_YSize ySize
0102 fc_Style style
0103 fc_Flags flags
0104
```

FontContentsHeader (librariesdiskfont.h)
FontContentsHeader (DiskFont) 226

```
0000 fch_FileID fileId
0002 fch_NumEntries numEntries
0004
```

FreeList (workbench/workbench.h)
FreeList (Workbench) 274

```
0000 fl_NumFree numFree
0002 fl_MemList memList
0010
```

Gadget (intuition/intuition.h)
Gadget (Intuition) 71

```
0000 NextGadget nextGadget
0004 LeftEdge leftEdge
0006 TopEdge topEdge
0008 Width width
000A Height height
000C Flags flags
000E Activation activation
0010 GadgetType gadgetType
0012 GadgetRender gadgetRender
0016 SelectRender selectRender
001A GadgetText gadgetText
001E MutualExclude
     mutualExclude
0022 SpecialInfo specialInfo
0026 GadgetID gadgetID
0028 UserData userData
002C
```

GadgetInfo (intuition/intuitionbase.h)
GadgetInfo (Intuition) 76

```
0000 gi_Environ environ
0004 gi_Gadget gadget
0008 gi_Box box
0010 gi_Container container
0018 gi_Layer layer
001C gi_NewKnob newKnob
0024
```

GamePortTrigger (devices/gameport.h)
GamePortTrigger (GamePort) 291

```
0000 gpt_Keys keys
0002 gpt_Timeout timeout
```

```
0004 gpt_XDelta xDelta
0006 gpt_YDelta yDelta
0008
```

GelsInfo (graphics/rastport.h)
GelsInfo (Graphics) 152

```
0000 sprRsrvd sprRsrvd
0001 Flags flags
0002 gelHead gelHead
0006 gelTail gelTail
000A nextLine nextLine
000E lastColor lastColor
0012 collHandler collHandler
0016 leftmost leftmost
0018 rightmost rightmost
001A topmost topmost
001C bottommost bottommost
001E firstBlissObj
     firstBlissObj
0022 lastBlissObj lastBlissObj
0026
```

GfxBase (graphics/gfxbase.h)
GfxBase (Graphics) 154

```
0000 LibNode libNode
0022 ActiView actiView
0026 copinit copinit
002A cia cia
002E blitter blitter
0032 LOFlist loFlist
0036 SHFlist shFlist
003A blthd blthd
003E bltt1 blttl
0042 bsblthd bsblthd
0046 bsblttl bsblttl
004A vbsrv vbsrv
0060 timsrv timsrv
0076 bltsrv. bltsrv
008C TextFonts textFonts
009A DefaultFont defaultFont
009E Modes modes
00A0 VBlank vBlank
00A1 Debug debug
00A2 BeamSync beamSync
00A4 system_bplcon0 bplcon0
00A6 SpriteReserved
spriteReserved
00A7 bytereserved
     bytereserved
00A8 Flags flags
00AA BlitLock blitLock
00AC BlitNest blitNest
00AE BlitWaitQ blitWaitQ
00BC BlitOwner blitOwner
00C0 TOF_WaitQ waitQ
00CE DisplayFlags
     displayFlags
00D0 SimpleSprites
     simpleSprites
00D4 MaxDisplayRow
     maxDisplayRow
00D6 MaxDisplayColumn
     maxDisplayColumn
00D8 NormalDisplayRows
     normalDisplayRows
00DA NormalDisplayColumns
     normalDisplayColumns
00DC NormalDPMX normalDPMX
00DE NormalDPMY normalDPMY
00E0 LastChanceMemory
     lastChanceMemory
00E4 LCMptr lcMptr
00E8 MicrosPerLine
     microsPerLine
00EA MinDisplayColumn reserved
00EC reserved reserved
0148
```

GListEnv (intuition/intuitionbase.h)
GListEnv (Intuition) 77

```
0000 ge_Screen screen
0004 ge_Window window
0008 ge_Requester requester
000C ge_RastPort rastPort
0010 ge_Layer layer
0014 ge_GZZLayer gzzLayer
0018 ge_Pens pens
001A ge_Domain domain
0022 ge_GZZdims gzzDims
002A
```

IBox (intuition/intuitionbase.h)
IBox (Intuition) 79

```
0000 Left left
0002 Top top
0004 Width width
0006 Height height
0008
```

Image (intuition/intuition.h)
Image (Intuition) 79

```
0000 LeftEdge leftEdge
0002 TopEdge topEdge
0004 Width width
0006 Height height
0008 Depth depth
000A ImageData imageData
000E PlanePick planePick
000F PlaneOnOff planeOnOff
0010 NextImage nextImage
0014
```

InfoData (libraries/dos.h)
InfoData (Dos) 213

```
0000 id_NumSoftErrors numSof-
tErrors
0004 id_UnitNumber unitNumber
0008 id_DiskState diskState
000C id_NumBlocks numBlocks
0010 id_NumBlocksUsed num-
BlocksUsed
0014 id_BytesPerBlock byte-
sPerBlock
0018 id_DiskType diskType
001C id_VolumeNode volumeNode
0020 id_InUse inUse
0024
```

InputEvent (devices/inputevent.h)
InputEvent (InputEvent) 332

```
0000 ie_NextEvent nextEvent
0004 ie_Class class
0005 ie_SubClass subClass
0006 ie_Code code
0008 ie_Qualifier qualifier
000A ie_x x
000C ie_y y
000A ie_addr eventAddress
000E ie_TimeStamp timeStamp
0016
```

Interrupt (exec/interrupts.h)
Interrupt (Exec) 32

```
0000 is_Node node
000E is_Data data
0012 is_Code code
0016
```

IntuiMessage (intuition/intuition.h)
IntuiMessage (Intuition) 81

```
0000 ExecMessage execMessage
0014 Class class
0018 Code code
001A Qualifier qualifier
001C IAddress iAddress
0020 MouseX mouseX
0022 MouseY mouseY
0024 Seconds seconds
0028 Micros micros
002C IDCMPWindow idcmpWindow
0030 specialLink SpecilaLink
0034
```

IntuiText (intuition/intuition.h)
IntuiText (Intuition) 82

```
0000 FrontPen frontPen
0001 BackPen backPen
0002 DrawMode drawMode
0004 LeftEdge leftEdge
0006 TopEdge topEdge
0008 ITextFont iTextFont
000C IText iText
0010 NextText nextText
0014
```

IntuitionBase (intuition/intuitionbase.h)
IntuitionBase (Intuition) 84

```
0000 LibNode libNode
0022 ViewLord viewLord
0034 ActiveWindow activeWindow
0038 ActiveScreen activeScreen
003C FirstScreen firstScreen
0040 Flags flags
0044 MouseY mouseY
0046 MouseX mouseX
0048 Seconds seconds
004C Micros micros
0050 MinXMouse minXMouse
0052 MaxXMouse maxXMouse
0054 MinYMouse minYMouse
0056 MaxYMouse maxYMouse
0058 StartSecs startSecs
005C StartMicros startMicros
0060 SysBase sysBase
0064 GfxBase gfxBase
0068 LayersBase layersBase
006C ConsoleDevice consoleDevice
```

```
0070 APointer aPointer
0074 APtrHeight aPtrHeight
0075 APtrWidth aPtrWidth
0076 AXOffset aXOffset
0077 AYOffset aYOffset
0078 MenuDrawn menuDrawn
007A MenuSelected menuSelected
007C OptionList optionList
007E MenuRPort menuRPort
00E2 MenuTmpRas menuTmpRas
00EA ItemCRect itemCRect
0112 SubCRect subCRect
013A IBitMap iBitMap
0162 SBitMap sBitMap
018A InputRequest inputRequest
01BA inputInterrupt
     InputInterrupt
01C0 EventKey eventKey
01C4 IEvents iEvents
01C8 EventCount eventCount
01CA IEBuffer ieBuffer
0232 ActiveGadget activeGadget
0236 ActivePInfo activePInfo
023A ActiveImage activeImage
023E GadgetEnv gadgetEnv
0268 GadgetInfo gadgetInfo
028C KnobOffset knobOffset
0290 getOKWindow getOKWindow
0294 getOKMessage getOKMessage
0298 setWExcept setWExcept
029A GadgetReturn gadgetReturn
029C StateReturn stateReturn
029E RP rp
02A2 ITmpRas iTmpRas
02AA OldClipRegion
     oldClipRegion
02AE OldScroll oldScroll
02B2 IFrame iFrame
02BA hthick hthick
02BC vthick vthick
02BE frameChange frameChange
02C2 sizeDrag sizeDrag
02C6 FirstPt firstPt
02CA OldPt oldPt
02CE SysGadgets sysGadgets
030E CheckImage checkImage
0316 AmigaIcon amigaIcon
031E apattern aPattern
032E bpattern bPattern
0336 IPointer iPointer
033A IPtrHeight iPtrHeight
033B IPtrWidth iPtrWidth
033C IXOffset iXOffset
033D IYOffset iYOffset
033E DoubleSeconds
     doubleSeconds
0342 DoubleMicros doubleMicros
0346 WBorLeft wBorLeft
0348 WBorTop wBorTop
034A WBorRight wBorRight
034C WBorBottom wBorBottom
034E BarVBorder barVBorder
0350 BarHBorder barHBorder
0352 MenuVBorder menuVBorder
0354 MenuHBorder menuHBorder
0356 color0 color0
0358 color1 color1
035A color2 color2
035C color3 color3
035E color17 color17
0360 color18 color18
0362 color19 color19
0364 SysFont sysFont
036C Preferences preferences
0370 Echoes echoes
0374 ViewInitX viewInitX
0376 ViewInitY viewInitY
0378 CursorDX cursorDX
037A CursorDY cursorDY
037C KeyMap keyMap
0380 MouseYMinimum
     mouseYMinimum
0382 ErrorX errorX
0384 ErrorY errorY
0386 IOExcess ioExcess
03AE HoldMinYMouse
     holdMinYMouse
03B0 WBPort wbPort
03B4 iqd_FNKUHDPort fnkuhdPort
03B8 WBMessage wbMessage
03EC HitScreen hitScreen
03F0 SimpleSprite simpleSprite
03F4 AttachedSSprite
     attachedSSprite
03F8 GotSprite1 gotSprite1
03FA SemaphoreList
     semaphoreList
0408 ISemaphore iSemaphore
054A MaxDisplayHeight
     maxDisplayHeight
054C MaxDisplayRow
     maxDisplayRow
054E MaxDisplayWidth
     maxDisplayWidth
0550 Reserved reserved
056E
```

D

IntVector (exec/interrupts.h)
IntVector (Exec) 32

```
0000 iv_Data data
0004 iv_Code code
0008 iv_Node node
000C
```

IOAudio (devices/audio.h)
IOAudio (Audio) 282

```
0000 ioa_Request request
0020 ioa_AllocKey allocKey
0022 ioa_Data data
0026 ioa_Length length
002A ioa_Period period
002C ioa_Volume volume
002E ioa_Cycles cycles
0030 ioa_WriteMsg writeMsg
0044
```

IOClipReq (devices/clipboard.h)
IOClipReq (Clipboard) 284

```
0000 io_Message message
0014 io_Device device
0018 io_Unit unit
001C io_Command command
001E io_Flags flags
001F io_Error error
0020 io_Actual actual
0024 io_Length length
0028 io_Data data
002C io_Offset offset
0030 io_ClipID clipID
0034
```

IODRPReq (devices/printer.h)
IODRPReq (Printer) 307

```
0000 io_Message message
0014 io_Device device
0018 io_Unit unit
001C io_Command command
001E io_Flags flags
001F io_Error error
0020 io_RastPort rastPort
0024 io_ColorMap colorMap
0028 io_Modes modesHi
              modes
002C io_SrcX srcX
002E io_SrcY srcY
0030 io_SrcWidth srcWidth
0032 io_SrcHeight srcHeight
0034 io_DestCols destCols
0038 io_DestRows destRows
003C io_Special special
003E
```

IOExtPar (devices/parallel.h)
IOExtPar (Parallel) 301

```
0000 IOPar ioPar
0030 io_PExtFlags pExtFlags
0034 io_Status status
0035 io_ParFlags parFlags
0036 io_PTermArray pTermArray
003E
```

IOExtSer (devices/serial.h)
IOExtSer (Serial) 312

```
0000 IOSer ioSer
0030 io_CtlChar ctlChar
0034 io_RBufLen rBufLen
0038 io_ExtFlags extFlags
003C io_Baud baud
0040 io_BrkTime brkTime
0044 io_TermArray termArray
004C io_ReadLen readLen
004D io_WriteLen writeLen
004E io_StopBits stopBits
004F io_SerFlags serFlags
0050 io_Status status
0052
```

IOExtTD (devices/trackdisk.h)
IOExtTD (TrackDisk) 321

```
0000 iotd_Req req
0030 iotd_Count count
0034 iotd_SecLabel secLabel
0038
```

IOPArray (devices/parallel.h)
IOPArray (Parallel) 302

```
0000 PTermArray0 pTermArray0
0004 PTermArray1 pTermArray1
0008
```

IOPrtCmdReq (devices/printer.h)
IOPrinter (Printer) 304

```
0000 io_Message message
0014 io_Device device
0018 io_Unit unit
001C io_Command command
001E io_Flags flags
001F io_Error error
0020 io_PrtCommand prtCommand
0022 io_Parm0 parm0
0023 io_Parm1 parm1
0024 io_Parm2 parm2
0025 io_Parm3 parm3
0026
```

IORequest (exec/io.h)
IORequest (Exec) 33

```
0000 io_Message message
0014 io_Device device
0018 io_Unit unit
001C io_Command command
001E io_Flags flags
001F io_Error error
0020
```

IOStdReq (exec/io.h)
IOStdReq (Exec) 35

```
0000 io_Message message
0014 io_Device device
0018 io_Unit unit
001C io_Command command
001E io_Flags flags
001F io_Error error
0020 io_Actual actual
0024 io_Length length
0028 io_Data data
002C io_Offset offset
0030
```

IOTArray (devices/serial.h)
IOTArray (Serial) 315

```
0000 TermArray0 termArray0
0004 TermArray1 termArray1
0008
```

Isrvstr (graphics/graphint.h)
Isrvstr (Graphics) 159

```
0000 is_Node node
000E Iptr iptr
0012 code code
0016 ccode ccode
001A Carg carg
001E
```

KeyMap (devices/keymap.h)
KeyMap (KeyMap) 336

```
0000 km_LoKeyMapTypes loKeyMapTypes
0004 km_LoKeyMap loKeyMap
0008 km_LoCapsable loCapsable
000C km_LoRepeatable loRepeatable
0010 km_HiKeyMapTypes
     hiKeyMapTypes
0014 km_HiKeyMap hiKeyMap
0018 km_HiCapsable hiCapsable
001C km_HiRepeatable
     hiRepeatable
0020
```

KeyMapNode (devices/keymap.h)
KeyMapNode (KeyMap) 338

```
0000 kn_Node node
000E kn_KeyMap keyMap
002E
```

KeyMapResource (devices/keymap.h)
KeyMapResource (KeyMap) 339

```
0000 kr_Node node
000E kr_List list
001C
```

Layer (graphics/clip.h)
Layer (Graphics) 160

```
0000 front front
0004 back back
0008 ClipRect clipRect
000C rp rp
0010 bounds bounds
0018 reserved reserved
001C priority priority
001E Flags flags
0020 SuperBitMap superBitMap
```

D

0022 SuperClipRect superClipRect
0026 Window window
002A Scroll_X scrollX
002C Scroll_Y scrollY
0030 cr cr
0034 cr2 cr2
0038 crnew crnew
003C SuperSaveClipRects superSaveClipRects
0040 _cliprects cliprects
0044 LayerInfo layerInfo
0048 Lock lock
0076 reserved3 reserved3
007E ClipRegion clipRegion
0082 saveClipRects saveClipRects
0086 reserved2 reserved2
009C DamageList damageList
00A0

Layer_Info (graphics/layers.h)
LayerInfo (Graphics) 164

0000 top_layer layer
0004 check_lp lp
0008 obs obs
000C FreeClipRects freeClipRects
0018 Lock lock
0046 gs_Head head
0054 longreserved longreserved
0058 Flags flags
005A fatten_count count
005B LockLayersCount lockLayersCount
005C LayerInfo_extra_size layerInfoExtraSize
005E blitbuff blitbuff
0062 LayerInfo_extra layerInfoExtra
0066

Library (exec/libraries.h)
Library (Exec) 37

0000 lib_Node node
000E lib_Flags flags
000F lib_pad pad
0010 lib_NegSize negSize
0012 lib_PosSize posSize
0014 lib_Version version
0016 lib_Revision revision
0018 lib_IdString idString
001C lib_Sum sum
0020 lib_OpenCnt openCnt
0022

List (exec/lists.h)
List (Exec) 38

0000 lh_Head head
0004 lh_Tail tail
0008 lh_TailPred tailPred
000C lh_Type type
000D lh_pad pad
000E

LoadSegBlock (devices/hardblocks.h)
LoadSegBlock (Hardblocks) 331

0000 lsb_ID id
0004 lsb_SummedLongs summed-Longs
0008 lsb_ChkSum chkSum
000C lsb_HostID hostID
0010 lsb_Next next
0014 lsb_LoadData loadData
0200

MemChunk (exec/memory.h)
MemChunk (Exec) 40

0000 mc_Next next
0004 mc_Bytes bytes
0008

MemEntry (exec/memory.h)
MemEntry (Exec) 41

0000 meu_Reqs reqs
0000 meu_Addr addr
0004 me_Length length
0008

MemHeader (exec/memory.h)
MemHeader (Exec) 42

0000 mh_Node node
000C mh_Attributes attributes
0010 mh_First first
0014 mh_Lower lower
0018 mh_Upper upper
001C mh_Free free
0020

MemList (exec/memory.h)
MemList (Exec) 43

```
0000 ml_Node node
000E ml_NumEntries numEntries
0010 ml_ME me
0018
```

Menu (intuition/intuition.h)
Menu (Intuition) 94

```
0000 NextMenu nextMenu
0004 LeftEdge leftEdge
0006 TopEdge topEdge
0008 Width width
000A Height height
000C Flags flags
000E MenuName menuName
0012 FirstItem firstItem
0016 JazzX jazzX
0018 JazzY jazzY
001A BeatX beatX
001C BeatY beatY
001E
```

MenuItem (intuition/intuition.h)
MenuItem (Intuition) 96

```
0000 NextItem nextItem
0004 LeftEdge leftEdge
0006 TopEdge topEdge
0008 Width width
000A Height height
000C Flags flags
000E MutualExclude mutualExclude
0012 ItemFill itemFill
0016 SelectFill selectFill
001A Command command
001C SubItem subItem
0020 NextSelect nextSelect
0022
```

Message (exec/ports.h)
Message (Exec) 44

```
0000 mn_Node node
000E mn_ReplyPort replyPort
0012 mn_Length length
0014
```

MinList (exec/lists.h)
MinList (Exec) 45

```
0000 mlh_Head head
0004 mlh_Tail tail
0008 mlh_TailPred tailPred
000C
```

MinNode (exec/nodes.h)
MinNode (Exec) 45

```
0000 mln_Succ succ
0004 mln_Pred pred
0008
```

MiscResource (resources/misc.h)
MiscResource (Resources) 270

```
0000 mr_Library library
0022 mr_AllocArray allocArray
0032
```

mouth_rb (devices/narrator.h)
Mouth (Narrator) 299

```
0000 voice voice
0046 width width
0047 height height
0048 shape shape
0049 pad pad
004A
```

MsgPort (exec/ports.h)
MsgPort (Exec) 46

```
0000 mp_Node node
000E mp_Flags flags
000F mp_SigBit sigBit
0010 mp_SigTask sigTask
0014 mp_MsgList msgList
0022
```

narrator_rb (devices/narrator.h)
IONarrator (Narrator) 297

```
0000 message message
0030 rate rate
0032 pitch pitch
0034 mode mode
0036 sex sex
```

```
0038 ch_masks chMasks
003C nm_masks nmMasks
003E volume volume
0040 sampfreq sampFreq
0042 mouths mouths
0043 chanmask chanMask
0044 numchan numChan
0045 pad pad
0046
```

NewScreen (intuition/intuition.h)
NewScreen (Intuition) 98

```
0000 LeftEdge leftEdge
0002 TopEdge topEdge
0004 Width width
0006 Height height
0008 Depth depth
000A DetailPen detailPen
000B BlockPen blockPen
000C ViewModes viewModes
000E Type type
0010 Font font
0014 DefaultTitle defaultTitle
0018 Gadgets gadgets
001C CustomBitMap customBitMap
0020
```

NewWindow (intuition/intuition.h)
NewWindow (Intuition) 100

```
0000 LeftEdge leftEdge
0002 TopEdge topEdge
0004 Width width
0006 Height height
0008 DetailPen detailPen
0009 BlockPen blockPen
000A IDCMPFlags idcmpFlags
000E Flags flags
0012 FirstGadget firstGadget
0016 CheckMark checkMark
001A Title title
001E Screen screen
0022 BitMap bitMap
0026 MinWidth minWidth
0028 MinHeight minHeight
002A MaxWidth maxWidth
002C MaxHeight maxHeight
002E Type type
0030
```

Node (exec/nodes.h)
Node (Exec) 48

```
0000 ln_Succ succ
0004 ln_Pred pred
0008 ln_Type type
0009 ln_Pri pri
000A ln_Name name
000E
```

PartitionBlock (devices/hardblocks.h)
PartitionBlock (Hardblocks) 340

```
0000 pb_ID id
0004 pb_SummedLongs summedLongs
0008 pb_ChkSum chkSum
000C pb_HostID hostId
0010 pb_Next next
0014 pb_Flags flags
0018 pb_Reserved1 reserved1
0020 pd_DevFlags devFlags
0024 pb_DriveName driveName
0044 pb_Reserved2 reserved2
004C pb_Enviroment enviroment
0090 pb_Reserved reserved
00CC
```

PenPair (intuition/intuitionbase.h)
PenPair (Intuition) 102

```
0000 DetailPen detailPen
0001 BlockPen blockPen
0002
```

Point (intuition/intuitionbase.h)
Point (Intuition) 103

```
0000 X x
0002 Y y
0004
```

Preferences (intuition/intuition.h)
Preferences (Intuition) 103

```
0000 FontHeight fontHeight
0001 PrinterPort printerPort
0002 BaudRate baudRate
0004 KeyRptSpeed keyRptSpeed
000C KeyRptDelay keyRptDelay
0014 DoubleClick doubleClick
```

```
001C PointerMatrix
     pointerMatrix
0064 XOffset xOffset
0065 YOffset yOffset
0066 color17 color17
0068 color18 color18
006A color19 color19
006C PointerTicks pointerTicks
006E color0 color0
0070 color1 color1
0072 color2 color2
0074 color3 color3
0076 ViewXOffset viewXOffset
0077 ViewYOffset viewYOffset
0078 ViewInitX viewInitX
007A ViewInitY viewInitY
007C EnableCLI enableCLI
007E PrinterType printerType
0080 PrinterFilename
     printerFilename
009E PrintPitch printPitch
00A0 PrintQuality printQuality
00A2 PrintSpacing printSpacing
00A4 PrintLeftMargin
     printLeftMargin
00A6 PrintRightMargin
     printRightMargin
00A8 PrintImage printImage
00AA PrintAspect printAspect
00AC PrintShade printShade
00AE PrintThreshold
     printThreshold
00B0 PaperSize paperSize
00B2 PaperLength paperLength
00B4 PaperType paperType
00B6 SerRWBits serRWBits
00B7 SerStopBuf serStopBuf
00B8 SerParShk serParShk
00B9 LaceWB laceWB
00BA WorkName workName
00D8 RowSizeChange padding
00D9 ColumnSizeChange
00DA PrintFlags
00DC PrintMaxWidth
00DE PrintMaxHeigth
00E0 PrintDensity
00E1 PrintXOffset
00E2 wb_Width
00E4 wb_Heigth
00E6 wb_Depth
00E7 ext_size
00E8
```

PrinterData (devices/prtbase.h)
PrinterData (PrtBase) 341

```
0000 pd_Device device
0034 pd_Unit unit
0056 pd_PrinterSegment
     printerSegment
005A pd_PrinterType
     printerType
005C pd_SegmentData
     segmentData
0060 pd_PrintBuf printBuf
0064 pd_PWrite pWrite
0068 pd_PBothReady pBothReady
006C pd_p0 p0
006C pd_s0 s0
00BE pd_p1 p1
00BE pd_s1 s1
0110 pd_TIOR tior
0138 pd_IORPort iorPort
015A pd_TC tc
01B6 pd_Stk stk
09B6 pd_Flags flags
09B7 pd_pad pad
09B8 pd_Preferences
     preferences
0AA0 pd_PWaitEnabled
     pWaitEnabled
0AA2
```

PrinterExtendedData (devices/prtbase.h)
PrinterExtendedData (PrtBase) 343

```
0000 ped_PrinterName
     printerName
0004 ped_Init init
0008 ped_Expunge expunge
000C ped_Open open
0010 ped_Close close
0014 ped_PrinterClass
     printerClass
0015 ped_ColorClass colorClass
0016 ped_MaxColumns maxColumns
0017 ped_NumCharSets
     numCharSets
0018 ped_NumRows numRows
001A ped_MaxXDots maxXDots
001E ped_MaxYDots maxYDots
0022 ped_XDotsInch xDotsInch
0024 ped_YDotsInch yDotsInch
0026 ped_Commands commands
002A ped_DoSpecial doSpecial
```

D

```
002E ped_Render render
0032 ped_TimeoutSecs
     timeoutSecs
0036 ped_8BitChars
     EightBitChars
003A ped_PrintMode
003E ped_ConvFunc
0042
```

PrinterSegment (devices/prtbase.h)
PrinterSegment (PrtBase) 347

```
0000 ps_NextSegment
     nextSegment
0004 ps_runAlert runAlert
0008 ps_Version version
000A ps_Revision revision
000C ps_ped ped
004E
```

Process (libraries/dosextens.h)
Process (Dos) 214

```
0000 pr_Task task
005C pr_MsgPort msgPort
007E pr_Pad pad
0080 pr_SegList segList
0084 pr_StackSize stackSize
0088 pr_GlobVec globVec
008C pr_TaskNum taskNum
0090 pr_StackBase stackBase
0094 pr_Result2 result2
0098 pr_CurrentDir currentDir
009C pr_CIS cis
00A0 pr_COS cos
00A4 pr_ConsoleTask
     consoleTask
00A8 pr_FileSystemTask
     fileSystemTask
00AC pr_CLI cli
00B0 pr_ReturnAddr returnAddr
00B4 pr_PktWait pktWait
00B8 pr_WindowPtr windowPtr
00BC
```

PropInfo (intuition/intuition.h)
PropInfo (Intuition) 111

```
0000 Flags flags
0002 HorizPot horizPot
0004 VertPot vertPot
0006 HorizBody horizBody
0008 VertBody vertBody
000A CWidth cWidth
000C CHeight cHeight
000E HPotRes hPotRes
0010 VPotRes vPotRes
0012 LeftBorder leftBorder
0014 TopBorder topBorder
0016
```

RasInfo (graphics/view.h)
RasInfo (Graphics) 165

```
0000 Next next
0004 BitMap bitMap
0008 RxOffset rxOffset
000A RyOffset ryOffset
000C
```

RastPort (graphics/rastport.h)
RastPort (Graphics) 166

```
0000 Layer layer
0004 BitMap bitMap
0008 AreaPtrn areaPtrn
000C TmpRas tmpRas
0010 AreaInfo areaInfo
0014 GelsInfo gelsInfo
0018 Mask mask
0019 FgPen fgPen
001A BgPen bgPen
001B AOlPen aOlPen
001C DrawMode drawMode
001D AreaPtSz areaPtSz
001E linpatcnt linPatCnt
001F dummy dummy
0020 Flags flags
0022 LinePtrn linePtrn
0024 cp_x x
0026 cp_y y
0028 minterms minterms
0030 PenWidth penWidth
0032 PenHeight penHeight
0034 Font font
0038 AlgoStyle algoStyle
0039 TxFlags txFlags
003A TxHeight txHeight
003C TxWidth txWidth
003E TxBaseline txBaseline
0040 TxSpacing txSpacing
0042 RP_User user
0046 longreserved longreserved
```

```
004E wordreserved wordreserved
005C reserved reserved
0064
```

Rectangle (graphics/gfx.h)
Rectangle (Graphics) 171

```
0000 MinX minX
0002 MinY minY
0004 MaxX maxX
0006 MaxY maxY
0008
```

Region (graphics/regions.h)
Region (Graphics) 172

```
0000 bounds bounds
0008 RegionRectangle
     regionRectangle
000C
```

RegionRectangle (graphics/regions.h)
RegionRectangle (Graphics) 173

```
0000 Next next
0004 Prev prev
0008 bounds bounds
0010
```

Remember (intuition/intuition.h)
Remember (Intuition) 113

```
0000 NextRemember nextRemember
0004 RememberSize rememberSize
0008 Memory memory
000C
```

Requester (intuition/intuition.h)
Requester (Intuition) 114

```
0000 OlderRequest olderRequest
0004 LeftEdge leftEdge
0006 TopEdge topEdge
0008 Width width
000A Height height
000C RelLeft relLeft
000E RelTop relTop
0010 ReqGadget reqGadget
0014 ReqBorder reqBorder
0018 ReqText reqText
001C Flags flags
001E BackFill backFill
0020 ReqLayer reqLayer
0024 ReqPad1 reqPad1
0044 ImageBMap imageBMap
0048 RWindow rWindow
004C ReqPad2 reqPad2
0070
```

Resident (exec/resident.h)
Resident (Exec) 49

```
0000 rt_MatchWord matchWord
0002 rt_MatchTag matchTag
0006 rt_EndSkip endSkip
000A rt_Flags flags
000B rt_Version version
000C rt_Type type
000D rt_Pri pri
000E rt_Name name
0012 rt_IdString idString
0016 rt_Init init
001A
```

RigidDiskBlock (devices/bootblock.h)
RigidDiskBlock (HardBlock) 348

```
0000 rdb_ID id:ARRAY [0..3] OF CHAR;
0004 rdb_SummedLongs:LONGCARD;
0008 rdb_ChkSum chkSum:LONGCARD;
000C rdb_HostID hostID:LONGCARD;
0010 rdb_Flags flags:RigidDiskFlagSet;
0014 rdb_BadBlockList:
     badBlockList:LONGCARD;
0018 rdb_PartitionList
     partitionList:LONGCARD;
001C rdb_FilesSysHeaderList
     filesSysHeaderList:LONGCARD;
0020 rdb_DriveInit driveInit:LONGCARD;
0024 rdb_Reserved1 reserved1:
     ARRAY [0..5] OF LONGCARD;
003C rdb_Cylinders cylinders:LONGCARD;
0040 rdb_Sectors sectors:LONGCARD;
0044 rdb_Heads heads:LONGCARD;
0048 rdb_Interleave interleave:LONGCARD;
004C rdb_Park park:LONGCARD;
0050 rdb_Reserved2 reserved2:
     ARRAY [0..2] OF LONGCARD;
005C rdb_WritePreComp
     writePreComp:LONGCARD;
0060 rdb_ReducedWrite
     reducedWrite:LONGCARD;
0064 rdb_StepRate
     stepRate:LONGCARD;
```

D

```
0068 rdb_Reserved3
     reserved3:ARRAY [0..4]
     OF LONGCARD;
007C rdb_RDBBlocksLo
     blocksLo:LONGCARD;
0080 rdb_RDBBlockHi
     blockHi:LONGCARD;
0084 rdb_LowCylinder
     lowCylinder:LONGCARD;
0088 rdb_HiCylinder
     hiCylinder:LONGCARD;
008C rdb_CylBlocks
     cylBlocks:LONGCARD;
0090 rdb_AutoParkSeconds
     autoParkSeconds:LONGCARD;
0094 rdb_Reserved4
     reserved4:ARRAY [0..1]
     OF LONGCARD;
009C rdb_DiskVendor
     diskVendor:ARRAY [0..7]
     OF CHAR;
00A4 rdb_DiskProduct
     diskProduct:ARRAY [0..15]
     OF CHAR;
00B4 rdb_DiskRevision
     diskRevision:ARRAY [0..3]
     OF CHAR;
00B8 rdb_ControllerVendor
     controllerVendor:
     ARRAY [0..7] OF CHAR;
00C0 rdb_ControllerProduct
     controllerProduct:
     ARRAY [0..15] OF CHAR;
00D0 rdb_ControllerRevision
     controllerRevision:
     ARRAY [0..3] OF CHAR;
00D4 rdb_Reserved5
     reserved5:ARRAY [0..9]
     OF LONGCARD;
00FC END;
```

RootNode (libraries/dosextens.h)
RootNode (Dos) 217

```
0000 rn_TaskArray taskArray
0004 rn_ConsoleSegment
     consoleSegment
0008 rn_Time time
0014 rn_RestartSeg restartSeg
0018 rn_Info info
001C rn_FileHandlerSegment
     fileHandlerSegment
0020
```

SatisfyMsg (devices/clipboard.h)
SatisfyMsg (Clipboard) 286

```
0000 sm_Msg msg
0014 sm_Unit unit
0016 sm_ClipID clipID
001A
```

Screen (intuition/intuition.h)
Screen (Intuition) 116

```
0000 NextScreen nextScreen
0004 FirstWindow firstWindow
0008 LeftEdge leftEdge
000A TopEdge topEdge
000C Width width
000E Height height
0010 MouseY mouseY
0012 MouseX mouseX
0014 Flags flags
0016 Title title
001A DefaultTitle defaultTitle
001E BarHeight barHeight
001F BarVBorder barVBorder
0020 BarHBorder barHBorder
0021 MenuVBorder menuVBorder
0022 MenuHBorder menuHBorder
0023 WBorTop wBorTop
0024 WBorLeft wBorLeft
0025 WBorRight wBorRight
0026 WBorBottom wBorBottom
0028 Font font
002C ViewPort viewPort
0054 RastPort rastPort
00B8 BitMap bitMap
00E0 LayerInfo layerInfo
0146 FirstGadget firstGadget
014A DetailPen detailPen
014B BlockPen blockPen
014C SaveColor0 saveColor0
014E BarLayer barLayer
0152 ExtData extData
0156 UserData userData
015A
```

Semaphore (exec/semaphores.h)
Semaphore (Exec)

```
0000 sm_MsgPort msgPort
0022 sm_Bids bids
0024
```

SemaphoreRequest (exec/semaphores.h)
SemaphoreRequest (Exec) 51

```
0000 sr_Link link
0008 sr_Waiter waiter
000C
```

SignalSemaphores (exec/semaphores.h)
SignalSemaphore (Exec) 52

```
0000 ss_Link link
000E ss_NestCount nestCount
0010 ss_WaitQueue waitQueue
001C ss_MultipleLink
     multipleLink
0028 ss_Owner owner
002C ss_QueueCount queueCount
002E
```

SimpleSprite (graphics/sprite.h)
SimpleSprite (Graphics) 173

```
0000 posctldata posctldata
0004 height height
0006 x x
0008 y y
000A num num
000C
```

SoftIntList (exec/interrupts.h)
SoftIntList (Exec) 53

```
0000 sh_List list
000E sh_Pad pad
0010
```

StandardPacket (libraries/dosextens.h)
StandardPacket (Dos) 218

```
0000 sp_Msg msg
0014 sp_Pkt pkt
0044
```

StringInfo (intuition/intuition.h)
StringInfo (Intuition) 119

```
0000 Buffer buffer
0004 UndoBuffer undoBuffer
0008 BufferPos bufferPos
000A MaxChars maxChars
000C DispPos dispPos
000E UndoPos undoPos
0010 NumChars numChars
0012 DispCount dispCount
0014 CLeft cLeft
0016 CTop cTop
0018 LayerPtr layerPtr
001C LongInt longInt
0020 AltKeyMap altKeyMap
0024
```

Task (exec/tasks.h)
Task (Exec) 54

```
0000 tc_Node node
000E tc_Flags flags
000F tc_State state
0010 tc_IDNestCnt idNestCnt
0011 tc_TDNestCnt tdNestCnt
0012 tc_SigAlloc sigAlloc
0016 tc_SigWait sigWait
001A tc_SigRecvd sigRecvd
001E tc_SigExcept sigExcept
0022 tc_TrapAlloc trapAlloc
0024 tc_TrapAble trapAble
0026 tc_ExceptData exceptData
002A tc_ExceptCode exceptCode
002E tc_TrapData trapData
0032 tc_TrapCode trapCode
0036 tc_SPReg spReg
003A tc_SPLower spLower
003E tc_SPUpper spUpper
0042 tc_Switch switch
0046 tc_Launch launch
004A tc_MemEntry memEntry
0058 tc_UserData userData
005C
```

TDU_PublicUnit (devices/trackdisk.h)
TDUPublicUnit (Trackisk) 322

```
0000 tdu_Unit unit
0026 tdu_Comp01Track
     comp01Track
0028 tdu_Comp10Track
     comp10Track
002A tdu_Comp11Track
     comp11Track
002C tdu_StepDelay stepDelay
0030 tdu_SettleDelay
     settleDelay
0034 tdu_RetryCnt retryCnt
0036
```

TextAttr (graphics/text.h)
TextAttr (Graphics) 175

```
0000 ta_Name name
0004 ta_YSize ySize
0006 ta_Style style
0007 ta_Flags flags
0008
```

TextFont (graphics/text.h)
TextFont (Graphics) 176

```
0000 tf_Message message
0014 tf_YSize ySize
0016 tf_Style style
0017 tf_Flags flags
0018 tf_XSize xSize
001A tf_Baseline baseline
001C tf_BoldSmear boldSmear
001E tf_Accessors accessors
0020 tf_LoChar loChar
0021 tf_HiChar hiChar
0022 tf_CharData charData
0026 tf_Modulo modulo
0028 tf_CharLoc charLoc
002C tf_CharSpace charSpace
0030 tf_CharKern charKern
0034
```

timerRequest (devices/timer.h)
timerRequest (Timer) 316

```
0000 tr_node node
0020 tr_time time
0028
```

timeval (devices/timer.h)
TimeVal (Timer) 317

```
0000 tv_secs secs
0004 tv_micro micro
0008
```

TmpRas (graphics/rastport.h)
TmpRas (Graphics) 178

```
0000 RasPtr rasPtr
0004 Size size
0008
```

UCopList (graphics/copper.h)
UCopList (Graphics) 179

```
0000 Next next
0004 FirstCopList firstCopList
0008 CopList copList
000C
```

Unit (exec/devices.h)
Unit (Exec) 57

```
0000 unit_MsgPort msgPort
0022 unit_flags flags
0023 unit_pad pad
0024 unit_OpenCnt openCnt
0026
```

View (graphics/view.h)
View (Graphics) 179

```
0000 ViewPort viewPort
0004 LOFCprList lofCprList
0008 SHFCprList shfCprList
000C DyOffset dyOffset
000E DxOffset dxOffset
0010 Modes modes
0012
```

ViewPort (graphics/view.h)
ViewPort (Graphics) 181

```
0000 Next next
0004 ColorMap colorMap
0008 DspIns dspIns
000C SprIns sprIns
0010 ClrIns clrIns
0014 UCopIns uCopIns
0018 DWidth dWidth
001A DHeight dHeight
001C DxOffset dxOffset
001E DyOffset dyOffset
0020 Modes modes
0022 SpritePriorities
0023 reserved reserved
0024
```

VSprite (graphics/gels.h)
VSprite (Graphics) 183

```
0000 NextVSprite nextVSprite
0004 PrevVSprite prevVSprite
0008 DrawPath drawPath
000C ClearPath clearPath
```

```
0010 OldY oldY
0012 OldX oldX
0014 Flags flags
0016 Y y
0018 X x
001A Height height
001C Width width
001E Depth depth
0020 MeMask meMask
0022 HitMask hitMask
0024 ImageData imageData
0028 BorderLine borderLine
002C CollMask collMask
0030 SprColors sprColors
0034 VSBob vsBob
0038 PlanePick planePick
0039 PlaneOnOff planeOnOff
003A VUserExt vUserExt
003C
```

WBArg (workbench/startup.h)
WBArg (Workbench) 275

```
0000 wa_Lock lock
0004 wa_Name name
0008
```

WBStartup (workbench/startup.h)
WBStartup (Workbench) 276

```
0000 sm_Message message
0014 sm_Process process
0018 sm_Segment segment
001C sm_NumArgs numArgs
0020 sm_ToolWindow toolWindow
0024 sm_ArgList argList
0028
```

Window (intuition/intuition.h)
Window (Intuition) 121

```
0000 NextWindow nextWindow
0004 LeftEdge leftEdge
0006 TopEdge topEdge
0008 Width width
000A Height height
000C MouseY mouseY
000E MouseX mouseX
0010 MinWidth minWidth
0012 MinHeight minHeight
0014 MaxWidth maxWidth
0016 MaxHeight maxHeight
0018 Flags flags
001C MenuStrip menuStrip
0020 Title title
0024 FirstRequest firstRequest
0028 DMRequest dmRequest
002C ReqCount reqCount
002E WScreen wScreen
0032 RPort rPort
0036 BorderLeft borderLeft
0037 BorderTop borderTop
0038 BorderRight borderRight
0039 BorderBottom borderBottom
003A BorderRPort borderRPort
003E FirstGadget firstGadget
0042 Parent parent
0046 Descendant descendant
004A Pointer pointer
004E PtrHeight ptrHeight
004F PtrWidth ptrWidth
0050 XOffset xOffset
0051 YOffset yOffset
0052 IDCMPFlags idcmpFlags
0056 UserPort userPort
005A WindowPort windowPort
005E MessageKey messageKey
0062 DetailPen detailPen
0063 BlockPen blockPen
0064 CheckMark checkMark
0068 ScreenTitle screenTitle
006C GZZMouseX gzzMouseX
006E GZZMouseY gzzMouseY
0070 GZZWidth gzzWidth
0072 GZZHeight gzzHeight
0074 ExtData extData
0078 UserData userData
007C WLayer wLayer
0080 IFont iFont
0084
```

E Alphabetisches Verzeichnis der Datenstrukturen

E

Amiga Datenstrukturen-Lexikon

»Amiga-Systemdatenstrukturen effektiv nutzen« – ein Begriff, der oft nur Geheimnisse beinhaltet. Grund dafür war bislang vor allem der Mangel an Fachliteratur, die das Thema »Datenstrukturen« nur nebensächlich behandelte.
Die vorhandene Lücke ist durch das vorliegende Buch geschlossen, denn es ist für alle Amiga-Programmierer geeignet, die systemspezifische Strukturen unabhängig von der Programmiersprache benutzen.
In diesem Lexikon werden alle Systemdatenstrukturen des Amiga unter Angabe der Offsets aufgelistet und ausführlich beschrieben. Dabei wird jeder Parameter umfassend erklärt.
Tabellarisch werden die Formate »C« und »Modula-2« verglichen. Diese beiden Programmiersprachen wurden bewußt gewählt: »C« kann als Standardsprache des Amiga bezeichnet werden, »Modula-2«, stellt die strengsten Anforderungen an die Syntax.
Aber auch die Basic- und Assembler-Freaks kommen auf ihre Kosten, und sei es nur durch die angegebenen Offsets.
Zu allen vier Programmiersprachen sind detaillierte Benutzerhinweise mit Anwendungsbeispielen beschrieben. Zusätzliche Referenzlisten der auf die jeweilige Datenstruktur zugreifenden Systemroutinen runden die einzelnen Beschreibungen ab.
Bei den ausführlichen Beschreibungen aller System-Datenstrukturen werden insbesondere folgende Aspekte berücksichtigt:

- das C-Format
- das INCLUDE-File
- das Modula-2-Format
- das Definitionsmodul
- der hexadezimale und dezimale Offset jeder Komponente
- die Größe der Datenstruktur in Byte, ebenfalls hexadezimal und dezimal
- Abweichungen des Assembler-Formats vom C-Format
- die Datenstrukturenreferenz, also eine Auflistung der Namen aller Datenstrukturen, die diese oder einen Adreßzeiger auf diese Datenstruktur beinhalten
- die Routinenreferenz, also eine Auflistung der Namen aller Systemroutinen, die auf diese Datenstruktur zugreifen
- eine Aufgabenbeschreibung der Datenstruktur
- eine Beschreibung jeder einzelnen Komponente der Datenstruktur

Hardware-Anforderungen:

Amiga 500, 1000, 2000 oder 2500

ISBN 3-89090-250-2

4 001057 902503

Markt&Technik

DM 69,–
sFr 66,20
öS 538,–